Sommario

SPECIALE

RUBRICHE

Direttore: Alessandro de Simone
Redazione/collaboratori: Claudio Baiocchi, Carlo e Lorenzo Barazzetta, Giovanni Bellù, Simone Bettola, Andrea e Alberto Boriani, Diego e Federico Canetta, Giancarlo Castagna, Umberto Colapicchioni, Pasquale D'Andreti, Maurizio Dell'Abate, Valerio Ferri, Luca Galluzzi, Michele Maggi, Giancarlo Mariani, Marco Miotti, Flavio Molinari, Claudio Mueller, Massimo Pollutri, Carla Rampi, Fabio Sorgato, Giovanni Verrelli, Antonio Visconti.
Segreteria di redazione: Maura Ceccaroli, Piera Perin
Ufficio Grafico: Mary Benvenuto, Arturo Ciaglia
Direzione, redazione, pubblicità: V.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
Pubblicità: Milano: Leandro Nencioni (direttore vendite), Giorgio Ruffoni, Roberto Sghirinzetti Claudio Tidone - V.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
- Emilia Romagna: Spazio E - P.zza Roosevelt, 4 - 40123 Bologna - Tel. 051/236979
- Toscana, Marche, Umbria: Mercurio Srl - via Rodari, 9 - San Giovanni Valdarno (Ar) - Tel. 055/947444
- Lazio, Campania: Spazio Nuovo - via P. Foscari 70 - 00139 Roma - Tel. 06/8109679

Segretaria: Marina Vantini - **Abbonamenti:** Paola Bertolotti
Tariffe: prezzo per copia L. 3.500. Abbonamento annuo (11 fascicoli) L. 35.000. Estero: il doppio. Abbonamento cumulativo alle riviste Computer e Commodore Computer Club L. 70.000.
I versamenti vanno indirizzati a: Systems Editoriale Srl mediante assegno bancario o utilizzando il c/c postale n. 37952207
Composizioni: Systems Editoriale Srl - **Fotolito:** Systems Editoriale Srl
Stampa: La Litografica S.r.l. - Busto Arsizio (VA)
Registrazione: Tribunale di Milano n. 370 del 2/10/82 - Direttore Responsabile: Michele Di Pisa
Sped. in abb. post. gr. III - Pubblicità inferiore al 70% - **Distrib:** MePe, via G. Carcano 32 - Milano

La rivoluzione d'ottobre

Senza aver la pretesa di innescare cambiamenti simili a quelli avvenuti in Russia all'inizio del secolo, noi, nel nostro piccolo...

di Alessandro de Simone

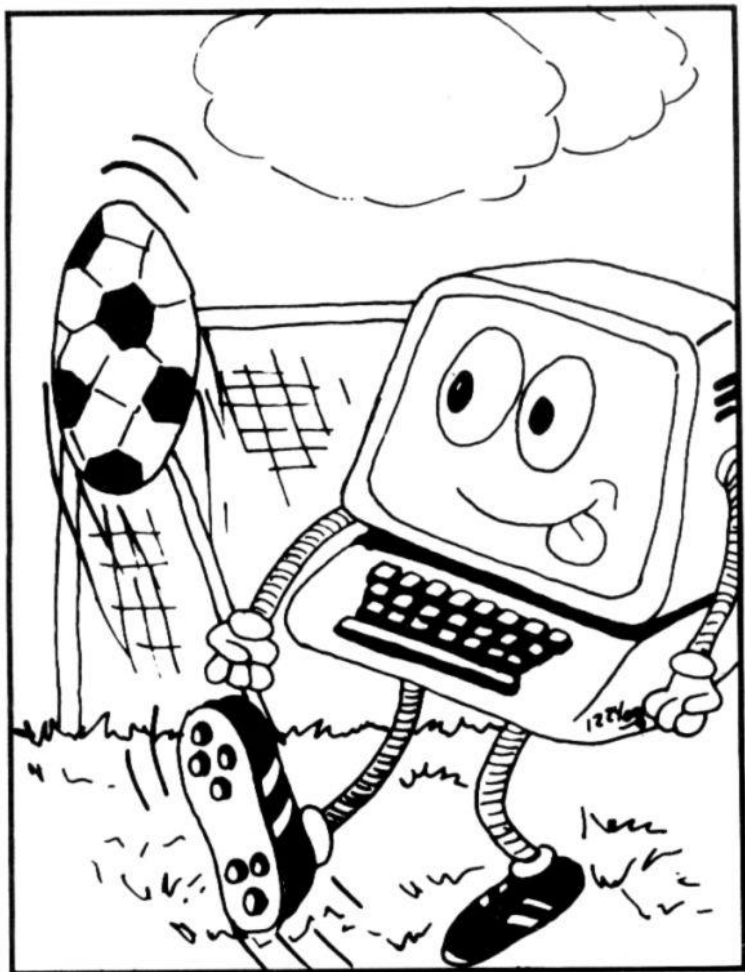
Molte telefonate sono giunte in Redazione in seguito all'iniziativa di pubblicare articoli sullo scottante tema delle protezioni.

Alcuni erano d'accordo, altri contrari, altri ancora si dichiaravano scettici sul futuro del software e non mancava chi, arrabbiato nero, metteva in evidenza l'enorme contraddizione della nostra rivista che da una parte pretende di ergersi a moralista e dall'altra accetta inserzionisti famosi per la pirateria che praticano.

Senza entrare nel merito delle osservazioni sull'argomento (che rimane sempre aperto) non possiamo evitare di mettere in evidenza le ultime due nuove iniziative che abbiamo deciso di intraprendere.

La prima riguarda, appunto, la diffusione del software a basso (issimo) prezzo che, a nostro parere, è l'unica arma valida contro la pirateria. A partire da questo mese, infatti, sarà possibile richiedere in Redazione un disco (su nastro ci è assolutamente impossibile) contenente, ogni mese, una quantità e qualità di software decisamente sproporzionate alla modesta cifra richiesta.

Rinviando alla parte del fascicolo che più da vicino descrive il nuovo servizio "Directory", occupiamoci della seconda iniziativa di cui desideriamo parlare.

Da una nostra indagine svolta presso alcune ricevitorie del Lotto (in occasione del famoso ritardo del 34) e del Totocalcio, durante le giornate

"critiche" del campionato, abbiamo osservato le cifre poderose che giocatori incalliti sono disposti a spendere nell'illusione di recuperarle ingigantite.

Ed ecco, quindi, la nostra idea: affrontare l'argomento dei giochi d'azzardo dimostrando, proprio col computer, che a vincere è sempre e solo l'organizzatore del gioco e quasi mai il giocatore, se consideriamo che, per ogni vincitore, DEVONO esservi decine di migliaia di perdenti.

Tratteremo programmi che, dimostrando scientificamente l'assurdità di riporre speranze su carte da gioco sparse sul tavolo verde o su una pallina che gira o su cavalli che (se dipendesse da loro) se ne starebbero in pace in un prato, facciano riflettere il potenziale giocatore sulla prudenza con cui è necessario affrontare il gioco, qualunque esso sia.

E' sempre presente, purtroppo, il rischio di ottenere l'effetto contrario, vale a dire spingere il lettore non-giocatore a rischiare cifre al Totocalcio, al Poker oppure tentare al Lotto.

Ma la fiducia nell'intelligenza dei nostri lettori ci sprona ad attendere listati, articoli e considerazioni da parte degli utenti di computer che, e sono tanti, la pensano come noi...

Accetti la sfida?

Gli autori del software pubblicato sul N.33 di C.C.C. ("Ecco gli sfidanti!" pag.19/24) che non hanno ancora ricevuto il materiale promesso in premio, sono pregati di comunicare in Redazione il loro indirizzo con estrema precisione.

Nel caso specifico della routine "Linee in bassa risoluzione", mi è sfuggito l'indirizzo presente nella riga 13470 e ne approfitto per chiedere scusa ai lettori che si fossero trovati in difficoltà a causa di questa mia omissione.

Errore in Enciclopedia di Routine

□ Sul N.31 (pag.93) la riga 13470 contiene un GoTo 10020 che però non esiste. A che serve quella riga dal momento che, nonostante tutto, il programma funziona egualmente?

• Alla riga 13470 si perviene dalla riga 13435 che ha il compito di verificare eventuali contraddizioni tra i parametri delle coordinate. In questo caso, infatti, i parametri stessi vengono alterati per evitare strani incidenti.

Il GoTo 10020 deve quindi esser modificato in GoTo 13420 per un corretto funzionamento. Nonostante l'errore riscontrato la routine funziona egualmente nel caso in cui non viene attivata la riga 13435, vale a dire nella quasi totalità dei casi.

Il motivo dell'errore riscontrato deriva dal fatto che i vari collaboratori, non potendo sapere quale sarà la numerazione finale che comparirà sulla rivista, inviano le varie routine numerate sempre da 10000 a 10099. In seguito, verificata la loro funzionalità ed idoneità alla pubblicazione, modifico le prime tre cifre di ciascuna riga (100) in quelle opportune (134 nel caso specifico). Uno dei vantaggi dell'enciclopedia di routine, grazie allo standard adottato, è proprio quello di poter renumerare a piacere le stesse routine modificando semplicemente le prime tre cifre delle linee e degli eventuali GoTo e GoSub presenti, lasciando inalterate le ultime due.

Repetita iuvant?

□ Affermate di non rispondere a domande già pubblicate, ma spesso ciò non avviene e rispondete egualmente affrontando argomenti già noti mentre trascurate, tra l'altro, le mie domande... ***(Filippo Gidoni - Mirano)***

• Quando un argomento viene richiesto nuovamente da un consistente numero di lettori non possiamo fare a meno di dedurre che vi sia una nuova "ondata" di acquirenti che si sono procurati la nostra rivista solo da poco tempo e che non possono, ovviamente, sapere se una risposta è stata già pubblicata.

Trascurare le loro legittime richieste sarebbe, quindi, ingiustificato.

Per ciò che riguarda la tua domanda, perchè non l'hai riproposta? Magari ti avrei risposto proprio in queste righe...

Vedo doppio

□ I computer Commodore possiedono due uscite video, una per il monitor ed una per il TV. E' possibile usarle contemporaneamente senza danni? ***(Giuseppe Pugliares - Augusta)***

• I circuiti cui accenni sono sempre attivi anche se non vengono utilizzati. E' pertanto possibile collegare sia un TV che un monitor senza pericoli di eventuali "sovraccarichi".

Tale sistema, anzi, è spesso usato negli stand di Fiere o in corsi pratici di computer quando si vuol far osservare lo schermo, oltre che all'operatore, anche ad un elevato numero di persone. L'operatore, infatti, lavora col monitor posizionato davanti alla tastiera mentre il TV, posto in alto in modo opportuno, consente una comoda osservazione ai partecipanti.

Stampare in grafica

□ Quali sono i comandi principali per stampare in alta risoluzione con la MPS/803? *(Carmelo Moraschiello, Foggia)*

• E' necessario ricorrere ad un "trucco" inviando particolari caratteri per far capire alla stampante l'intenzione di programmare caratteri personalizzati.

L'argomento è piuttosto lungo e (relativamente) complesso, ma ben spiegato nel manuale di istruzione della macchina e di tutte le stampanti 803/compatibili.

Modem, modem e ancora modem

□ Numerosissimi sono ormai i lettori che chiedono informazioni su quali modem acquistare, come collegarli, quali sono le banche dati, quali problemi potrebbero sorgere con la SIP.

Stiamo preparando un dossier sull'argomento che verrà pubblicato quanto prima.

Computer non funzionanti

□ Molti lettori (tra cui: R. Merozzi, F. Valendino) chiedono come fare per ripristinare il funzionamento di computer che presentano anomalie Hardware come, ad esempio, registratori che non girano, schermi "spenti", tastiere perennemente disabilitate ed altro). Alcuni inconvenienti, tra l'altro, risultano successivi a interventi pasticcioni con il saldatore.

• Non è assolutamente possibile per noi, come per chiunque altro, individuare il guasto in base ad una semplice descrizione del difetto nè, tantomeno, suggerirne il rimedio.

L'unica cosa da fare è quella di inviare la macchina presso un centro di riparazione e assistenza. Tra le nostre pagine vi sono i nominativi di alcuni inserzionisti che effettuano il servizio citato.

Linguaggi alieni

☐ **Sono disponibili per il Commodore 64 cartucce che utilizzino un Basic diverso dallo standard V2, e come procurarsele in caso affermativo?** *(Andrea Cascio, Livorno)*

• Per il C/64 sono in commercio, da parecchio tempo, decine di centinaia di migliaia di milioni di Basic e/o altri linguaggi (Pascal, PL/0, Turtle, Logo, Forth ed un'altra dozzina di paurosilioni di alternative).

Il sistema più semplice per procurarseli è quella di recarsi al negozio specializzato più vicino, oppure di contattare i lettori della rubrica "Scambiatevi le liste". Tra questi, infatti, moltissimi vantano una biblioteca di centinaia di programmi per i quali c'è solo l'imbarazzo della scelta.

Televideo by C/64?

☐ **Ho sentito dire che la scheda Televideo non è altro che la piastra del C/64. Se ciò non è vero, quale è il computer che consente il collegamento con Televideo?** *(Davide Crivellotto, Mediglia)*

• Il servizio Televideo consiste in un gruppo di informazioni che viene trasmesso, insieme con gli stessi programmi RAI, e che perviene, quindi, all'interno di qualsiasi TV domestico. Naturalmente tale "contemporaneità" non arreca alcun fastidio alla normale ricezione.

La parte "invisibile" può essere filtrata, estratta e visualizzata grazie ad una particolare scheda elettronica (Televideo, appunto), governata da un microprocessore specifico, capace di compiere l'operazione.

Tale scheda, benchè piuttosto particolare e in grado di svolgere un compito preciso ma limitato, risulta semplice dal momento che vi sono montati, in genere, solo quattro chip e pochi altri elementi. Se le case costruttrici inserissero all'interno dei TV prodotti un intero C/64 (tra l'altro difficilmente programmabile per lo scopo), lo stesso apparecchio costerebbe moltissimo e nessuno lo comprerebbe.

In conclusione: è vero che per il televideo c'è bisogno di una piastra a microprocessore, ma è anche vero che questa non è un computer in commercio, ma una scheda elettronica appositamente costruita.

Accoppiamenti particolari

☐ **Esiste un'interfaccia che consente di collegare il C/64 con una macchina da scrivere elettronica Olivetti?** *(Massimiliano Lodi, Sarezzo)*

• Il C/64 dispone di vari tipi di interfaccia. Escludendo quelle incorporate (seriale per il drive e seriale per il registratore) ed immediatamente disponibili solo per apparecchi Commodore, è possibile utilizzare interfacce esterne per inserire il C/64 su altri "bus" standardizzati e molto diffusi.

Tra questi ricordiamo, principalmente, l'interfaccia seriale RS-232, e le parallele IEEE-488 e Centronics.

Se, dunque, la macchina da scrivere (o stampante) possiede uno dei tre connettori citati, il collegamento è possibile, almeno in teoria.

Se, infatti, si vuole adoperare la macchina da scrivere con programmi professionali (W/P, Data Base, Spreadsheet ed altri) non sempre il collegamento sembra funzionare. Ciò è dovuto al fatto che tali programmi non hanno previsto la possibilità di uscita su altri bus oppure, a causa di alcune protezioni, alterano il normale funzionamento dell'invio dati verso la stampante.

Prima dell'acquisto, pertanto è bene provare il computer collegato alla macchina da scrivere e verificarne il funzionamento caricando e lanciando i programmi che si intendono usare spesso.

Per procurarsi le varie interfacce ti suggeriamo di contattare i nostri inserzionisti che effettuano vendita per corrispondenza.

Prendi due, vedi uno

☐ **Si può sovrapporre la videata di un computer su quella di una normale trasmissione televisiva?** *(Mauro Melchionda, Sannicandro)*

• E' necessario un circuito che si chiama "miscelatore video" che, normalmente, è inserito nei processori video; questi sono apparecchi che iniziano ad avere grande diffusione grazie al successo di vendita dei videoregistratori: consentono, infatti, di miscelare le immagini provenienti da più apparecchi video (V/R, telecamere, computer) e risultano indispensabili per il cosiddetto "montaggio".

I processori video hanno un prezzo al pubblico che parte da mezzo milione di lire circa (in su...) e necessitano, per un funzionamento ottimale, di più apparecchi: monitor, video registratori, telecamere, titolatrici elettroniche eccetera.

Io cerco la TILina

☐ **Sul N.30 di C.C.C. indicate, tra i componenti per realizzare la penna ottica, il TIL 78 che risulta introvabile. Dove acquistarlo?** *(Manuel Ceretti, Biella)*

• Nei nostri progetti Hardware teniamo conto dei problemi che possono avere i lettori nel procurarsi il materiale ed utilizziamo, pertanto, com-

ponenti di facile reperibilità. Alcuni di essi, però, sono insostituibili per un perfetto risultato.

Il TIL 78 (o suo equivalente) è relativamente facile da trovare nei negozi specializzati e, nel malaugurato caso non si riesca a rintracciarli nella propria cittadina, è necessario (purtroppo) prendere il treno e recarsi in un centro più grande in cui operano negozi più forniti.

Hai provato, però, a contattare gli stessi autori dell'articolo? (Logical Instruments Tel.02/3511871).

Programmi protetti

☐ **Come mai alcuni programmi su nastro, perfettamente listabili, caricati su disco non funzionano?** *(Pietro Nicola, Guardiagrele)*

• Le protezioni agiscono, prevalentemente, in due modi: il primo impedisce la copia ed ogni tentativo genera vari errori (tra cui "Out of Memory"). Il secondo, più subdolo, fa credere che la copia sia possibile ma, agli atti pratici, il programma riprodotto non funziona, oppure funziona male. Mi pare che sia quest'ultimo il caso che ti crea problemi.

Moda o razionalità?

☐ **Ho notato che in commercio sono presenti due modelli di drive che differiscono nello sportello per l'introduzione del dischetto. Vi sono altre differenze?** *(Claudio Del Bianco, Firenze)*

• No: il motivo della differenza è da ricercarsi nella (probabile) economia realizzabile in fase di produzione in serie.

Chissà, comunque, che la Commodore non si sia voluta affiancare alla moda della "levetta" abbandonando lo sportello, forse troppo demodè...

Abbasso l'Azimuth

☐ **Perchè i registratori non li vendono con l'azimuth regolato in fabbrica e bloccato, una volta per tutte, all'altezza ottimale?** *(Luca Bertin, Padova)*

• In fabbrica la testina viene regolata alla giusta posizione. Problemi di trasporto e cattiva manutenzione da parte dell'utente (leggi: urti, vibrazioni eccetera) riescono tuttavia a spostare la testina dalla sua posizione originaria. Nel caso dei computer sono sufficienti pochi decimi di millimetro per perdere byte durante il trasferimento dei dati.

L'opportuno forellino che consente l'accesso alla vite di regolazione consente, quindi, una manutenzione facilissima, altrimenti impossibile.

L'ultima spiaggia

☐ **Come si fa a conoscere l'ultima locazione di un programma in linguaggio macchina?** *(Antonio Parziale, Brindisi)*

• Dopo averlo caricato da nastro (oppure da disco con la sintassi: LOAD"NOME",8,1), è sufficiente digitare:

PRINT PEEK(45)+PEEK(46)*256

Il numero che viene fuori rappresenta il valore cercato.

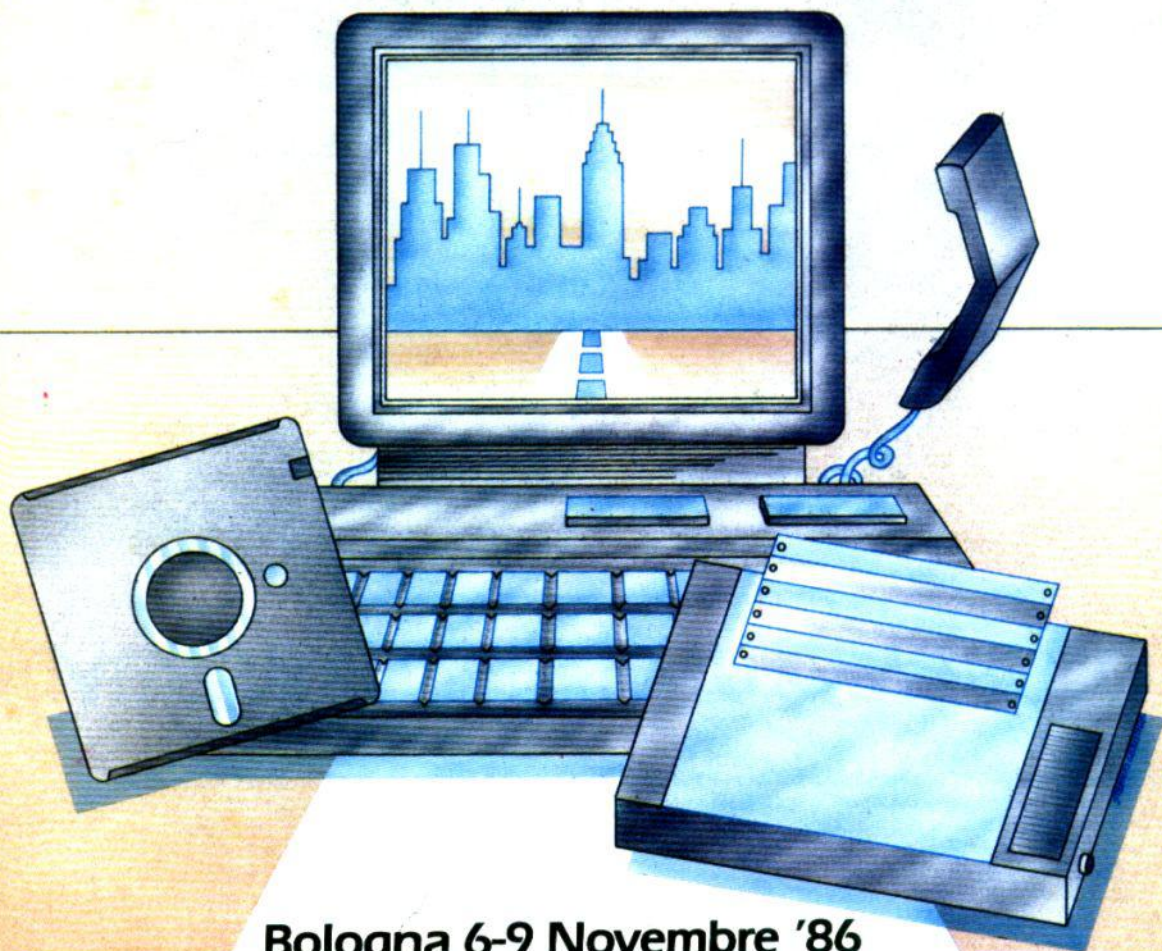
Errori

☐ **Il terzo listato "Labirinto" pubblicato sul N.29 (pag 71), genera un "Illegal quantity error in 580". Dov'è l'errore?** *(Enio Ducci, Firenze)*

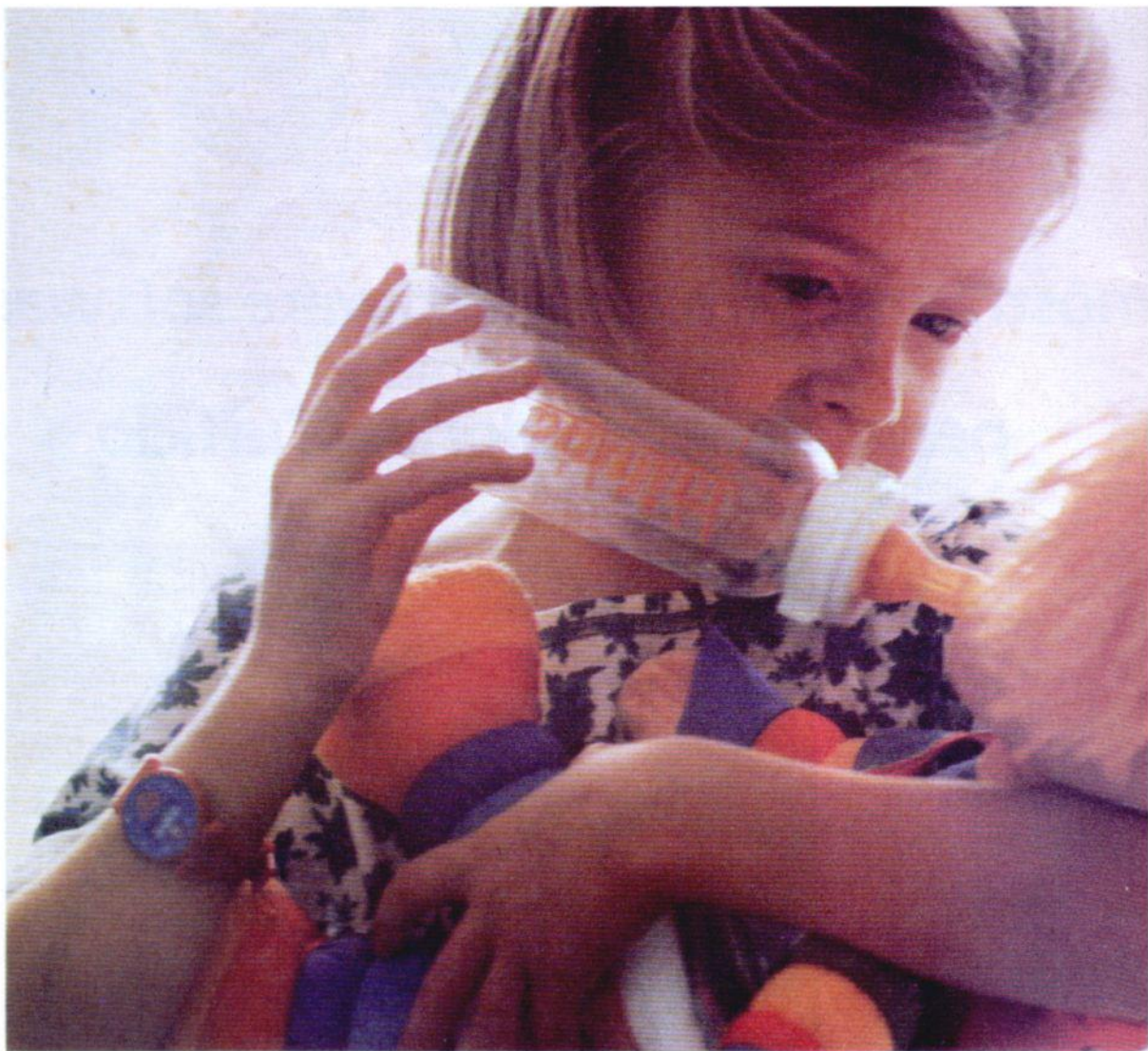
• Nella trascrizone che hai effettuato! Torniamo a ripetere che i nostri listati, tranne rarissime eccezioni di cui diamo immediata notizia, sono provati più volte prima della pubblicazione e non contengono, quindi, errori di sorta.

In generale: verifica riga per riga la perfetta corrispondenza tra il listato pubblicato e quello trascritto NON chiedendo un gruppo di righe alla volta (tipo: LIST 100-200) ma SOLTANTO una riga alla volta (Esempio: LIST 100; poi LIST 110, e così via).

Consigliamo, soprattutto, di leggere gli inserti dei numeri 31 e 32 che sono stati scritti proprio per spiegare in che modo agire quando vengono visualizzate segnalazioni di errore non previste.

mostra mercato dell'hardware software
e tecnologie
per la comunicazione e l'ufficio

Bologna 6-9 Novembre '86
Palazzo dei Congressi (Fiera)
orario mostra 10 - 19

Segreteria organizzativa: **PROMO EXPO** - Via Barberia, 22 - 40123 Bologna - Tel. (051) 33.36.57-33.27.42

Foto Gianni Baggi

DIAMO UNA MANO ALLA VITA,

Unicef è il Fondo delle Nazioni Unite per l'infanzia.

Creato nel 1946 per soccorrere i bambini vittime del secondo conflitto mondiale, ora si occupa esclusivamente dei paesi in via di sviluppo.

Oggi l'Unicef opera in 117 paesi del Terzo Mondo con l'obiettivo primario di dimezzare il tasso di mortalità infantile salvando 7 milioni di piccole vite all'anno e proteggere la salute e la crescita di molti milioni di altri bambini.

L'Unicef è apolitico e i suoi finanziamenti provengono esclusivamente dai contributi volontari.
Il 75% dei fondi provengono da stanziamenti governativi, mentre il 25% proviene da privati.

Allattamento al seno: il migliore alimento possibile.

Uno dei programmi dell'Unicef è la diffusione di metodi relativamente semplici e a basso costo per consentire agli stessi genitori di ridurre della metà il tasso di mortalità dei loro bambini e di salvarne fino a 20.000 al giorno.

Uno di questi è l'allattamento al seno. Esso garantisce ai bambini il migliore alimento possibile, unitamente ad un grado elevato di immunizzazione contro le malattie infettive durante i primi mesi di vita.
I bambini nutriti al seno hanno uno sviluppo più armonioso e maggiori possibilità di sopravvivere.

L'impegno dell'Unicef è quello di formare la maggiore

DOVE NUTRIRSI NON È UN GIOCO.

quantità possibile di personale sanitario di base per la diffusione di questi metodi, e lo sforzo finanziario è elevato. Anche tu puoi fare molto per risolvere questo problema.

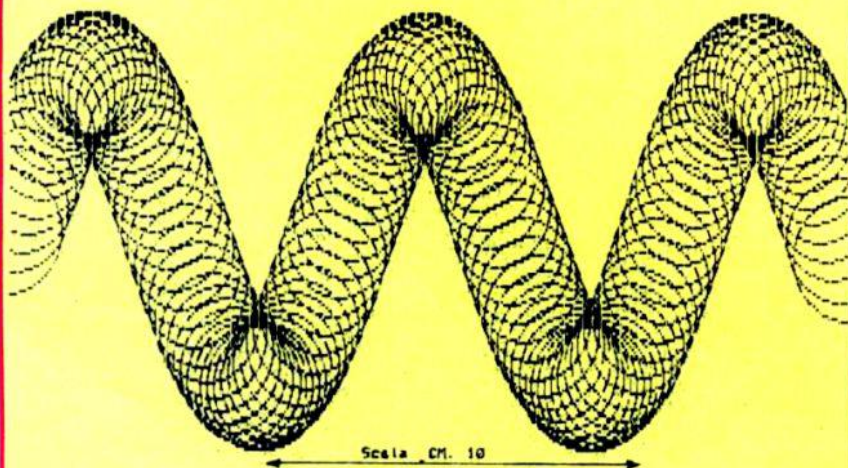
Milioni di bambini da aiutare sono buone ragioni per aiutare l'Unicef.

Puoi inviare il tuo contributo direttamente al Comitato Italiano per l'Unicef sul c/c postale n. 26479006, piazza Marconi 25, 00144 Roma. Grazie.
Per informazioni, cerca nell'elenco telefonico della tua città alla voce Unicef.

Unicef - 1946/1986 - Quarant'anni al servizio delle madri e dei bambini di tutto il mondo.

Graphic Expander per C/128 in 80 colonne

Un'espansione S/W a basso prezzo per gli utenti del potente computer grafico

di Giorgio Chiozzi

Nel settembre scorso, quando allo SMAU '85 la Commodore presentò ufficialmente al pubblico italiano il "nuovo" Commodore 128, vi era una diffusa sete di notizie, specialmente tra il pubblico più giovane, che non vedeva l'ora di esaminarlo da vicino.

Come sempre avviene in tali occasioni, ad ogni notizia fanno eco commenti, note di approvazione o severi rimproveri nei confronti della Casa costruttrice. Nel caso particolare del C/128, grande stupore suscitava la duplice uscita video, composta, come tutti oramai ben sanno, da un video 40 colonne e da uno da 80.

Tale stupore era accompagnato dalla perplessità, che fortunatamente si sarebbe rivelata infondata, dovuta all'impossibilità di collegare l'uscita 80 colonne ad un normale monitor composito, come quelli comunemente usati per il C/64.

Allo SMAU conobbi il direttore della Precision Software, Mr. Turner, un simpaticissimo signore sulla cinquantina al quale capitò la sventura di accettare un passaggio dal sottoscritto (con 500 gialla), dalla Fiera fino a Cinisello (!). Durante il viaggio, che "grazie" all'intenso traffico durò oltre due ore, cantando "We all live in a yellow submarine", Mr. Turner sottolineava pregi e difetti del C/128, computer che all'epoca conosceva molto meglio di noi standisti, continuamente costretti a salti mortali per soddisfare le cervellotiche domande del pubblico.

La mancanza che Mr. Turner considerava più grave era costituita dall'impossibilità di utilizzare la grafica sull'uscita a 80 colonne che, per definizione e qualità, è l'uscita video più interessante.

Incoraggiato da Mr. Turner e dalla disponibilità a fornire le informazioni necessarie dimostrata da parte di Diego Perini (ragazzo di 24 anni che sa tutto sui disk driver e che all'epoca lavorava per la Commodore), cominciai a scrivere un gruppo di routine grafiche per l'uscita ad 80 colonne.

La grafica nel C/128

La gestione dello schermo a 80 colonne è compito del'8563, processore video capace di un modo testo da 80 colonne per 25 righe e di un modo grafico di 640x200 punti; modo che può comunque essere alterato, lavorando sui numerosi e versatili registri di cui dispone.

L'8563 lavora su un banco display Ram (o meglio: Dram) di 16 kilobyte, non accessibili direttamente dalla Cpu 8510 che, per poter leggere o scrivere in una locazione Dram, è costretta ad un protocollo stranissimo, che vede il processore 8563 come insormontabile e "burocratico" intermediario.

L'estensione Basic

Graphic Expander 128 è un'estensione Basic che rende possibile, tramite comandi analoghi a quelli destinati alla grafica 320x200, l'utilizzo grafico dell'8563. L'utente può disegnare funzioni, istogrammi e diagrammi a torta usufruendo della professionale grafica 640x200. Sono inoltre previsti comandi che rendono possibile l'accesso da Basic al banco di memoria da 16 Kbyte destinato all'8563, ed un comando che consente di trasportare lo schermo grafico gestito dal VIC sullo schermo 640x200 gestito dal nuovo processore.

I nuovi comandi Basic

Passiamo ad una veloce rassegna dei comandi che Graphic Expander aggiunge al Basic "ufficiale" del C/128:

• Hcopy: esegue una hardcopy dello schermo 640x200 su stampante 801, 803 e compatibili. Il disegno, a causa della maggior dimensione rispetto ad un 40 colonne, viene stampato in verticale, riempendo quasi interamente un foglio di formato A4 (dimensioni output disegno: cm.15x23).

• Hgraphic grpmode: se grpmode=1 imposta il modo grafico 640x200, se grpmode=0 carica in Dram il set di caratteri 80 colonne, (il set caratteri dell'8563 è infatti memorizzato in Dram, ed ogni volta che si usa il bitmapping è necessario cancellarlo), ed attiva il modo testo.

• Hcolor frgnd,bkgnd: imposta il colore di scrittura (frgnd) ed il colore di sfondo (bkgnd).

• Hload: trasferisce lo schermo grafico 320x200 sullo schermo 640x200; va notato che tale comando espande i pixels dello schermo 320x200 affinchè la mappa grafica visualizzata sullo schermo grafico 640x200 mantenga le proporzioni originali.

• Hcircle pltmode, cx,cy,r: disegna un cerchio di centro cx,cy e raggio r; se pltmode=1 scrive, se pltmode=2 inverte, se pltmode=0 cancella.

• Hdraw pltmode,x,y [to x1,y1 [to x2,y2 ... [to xn,yn]]]: se vengono omessi i parametri opzionali, interni alle parentesi quadre, viene disegnato il punto di coordinate x,y; pltmode ha la funzione di far scrivere, invertire o cancellare; se sono presenti parametri opzionali vengono disegnate le n linee di cui vengono assegnate le coordinate degli estremi x,y x1,y1 x2,y2 .... xn,yn

• Hbox pltmode,x,y to x1,y1: disegna il rettangolo di cui sono assegnati i due spigoli x,y e x1,y1

• Hpaint pltmode,x,y: colora l'area di cui viene assegnato il punto interno x,y; se pltmode=0 l'area viene cancellata.

• Hpoke addr,byte: esegue una poke sul banco Dram dell'8563. Da notare che i due bit più significativi dell'indirizzo addr non vengono considerati: infatti i due pin "più significativi" del bus indirizzi dell'8563, che potrebbe indirizzare 64 kbyte Dram, sono addirittura scollegati sul circuito stampato del calcolatore!

• Hpeek (addr): legge una locazione del banco Dram dell'8563 e ne ritorna il valore.

• Hrdot (x,y): ritorna lo stato del pixel di coordinate x,y: 1=acceso, 0=spento.

• Hrgr (reg): legge il registro reg del processore video e ne ritorna il valore.

• Hrgr reg,byte: modifica il registro reg dell'8563, assegnandogli il valore byte.

• Hscnclr: pulisce la pagina grafica 640x200; va notato che questo comando cancella il set di caratteri 80 colonne, che normalmente è memorizzato nel banco Dram (16 kbyte) di cui necessita la grafica 640x200.

Come procurarsi Graphic Expander C/128

Assieme a Graphic Expander 128 viene fornito un programma dimostrativo che illustra le capacità del Basic ampliato e costituisce un esempio di come sia possibile utilizzare i nuovi comandi.
Sullo stesso supporto magnetico viene inoltre fornito un editor di caratteri, che consente la definizione di set di caratteri personalizzati.

I lettori interessati ad entrare in possesso del software in oggetto (fornito esclusivamente su disco) devono inviare la modica cifra di L.27000, comprensiva delle spese di spedizione. Non ci è possibile, infatti, inviare materiale contrassegno.

Compilate un normale modulo di C/C postale indirizzando a:

C/C postale N.37952207
Systems Editoriale
Viale Famagosta, 75
20142 Milano

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento") non solo il vostro nominativo completo, ma anche il nome del software desiderato: Graphic Expander C/128 (disco).

La routine "LIST": come funziona e come modificarla

Un gruppo di considerazioni che, tra l'altro, consentono di rimuovere un particolare tipo di protezione

di Roberto Morassi

Vi siete mai chiesti che cosa c'è "dietro" un semplice comando di LIST?

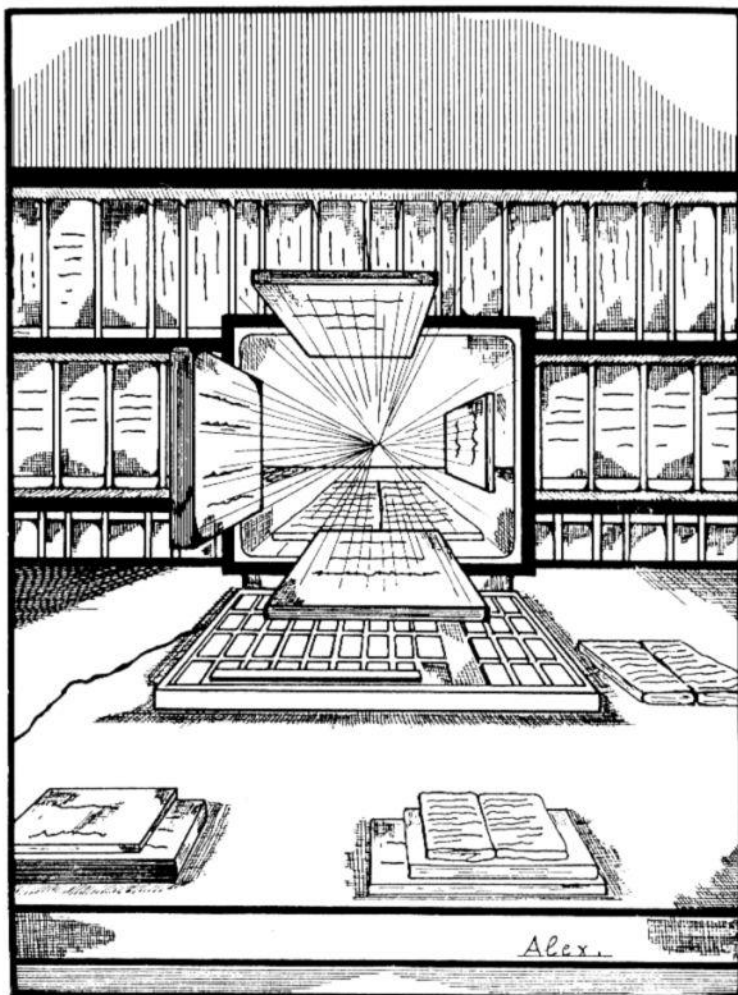
Per capire come funziona esaminiamo una per una le numerose routine che il Sistema Operativo del computer esegue quando si comanda il LIST.

Il suo punto di ingresso nel S.O è $A69C (decimale 42652). La prima parte della routine, come per altri comadi Basic, è la raccolta dei parametri dal testo: vengono cioè letti i valori che seguono eventualmente l'istruzione per poter fissare i limiti inferiore e superiore del listato.

In assenza di parametri, tali limiti vengono fissati rispettivamente all'inizio del testo Basic e alla riga (fittizia) 65535. In caso contrario viene richiamata una subroutine a $A613 che ricerca le locazioni a cui iniziano i numeri di riga richiesti.

Il limite superiore del listato viene copiato nel puntatore 20-21; in seguito inizia il LIST vero e proprio.

Il S.O. dapprima controlla se è giunto alla fine del programma Basic, altrimenti legge il numero di riga "corrente" e lo stampa se è minore o uguale al limite superiore del listato. Se è maggiore, il LIST viene concluso.

Subito dopo il numero di riga viene inserito uno spazio e prelevato il carattere successivo dal testo Basic (il registro Y serve da puntatore per scorrere lungo la linea Basic). Il passo successivo (analisi del carattere prelevato) è vettorizzato: passa cioè

attraverso un puntatore in zona RAM (registri 774-775) che normalmente punta a $A71A.

Quest'ultima routine esamina dapprima se il codice del carattere corrente è minore di 128 (bit 7 settato) e in tal caso lo stampa. Se è maggiore di 128 ma si trova in "modo stringa" viene ugualmente stampato; se infine è maggiore di 128 e non in "modo stringa", si tratta di un'istruzione Basic "tokeñizzata" (cioè codificata in un singolo byte) che viene decifrata e stampata per esteso.

Dopo la visualizzazione, il registro Y viene incrementato, e viene stampato con la stessa procedura il carattere successivo. Quando però si trova uno "zero" (segnale di fine linea), si passa alla linea seguente e si ripete nuovamente l'intero ciclo.

Un modo di proteggere

Tutti sapranno che è possibile inserire subito dopo il comando REM alcuni caratteri particolari, che non vengono ovviamente letti durante il RUN ma vengono interpretati e "stampati" dal LIST, spesso con strani risultati.

Provate ad esempio REM Shift-A: questo verrà letto come istruzione Basic "tokenizzata" e il listato darà REM ATN.

Altri trucchi ben noti sono l'introduzione di Shift-L, che blocca il listato con ?SYNTAX ERROR, e del Reverse-T (delete) che cancella il carattere che lo precede.

Ecco un interessante metodo per introdurre altri comandi nelle REM. Dopo REM, premete nell'ordine:

Due doppi apici, tasto Delete, Rvs-On, Shift-M, Shift-Delete, un tasto colore a scelta (ad esempio: tasto Control e tasto 1), e infine il tasto Return.

Da quel punto, il listato proseguirà nel colore che avrete scelto. Potete utilizzare i sedici colori disponibili per ottenere un listato in Technicolor!

Un altro metodo per alterare il LIST è quello di modificare il puntatore 774-775. Per esempio con POKE774,200 (o altri valori) si devia il LIST... chissà dove, e quindi, di fatto, lo impedisce. Per ripristinarlo, dovete riportare il puntatore al valore "default" con POKE774,26: POKE 775, 167 (o più semplicemente con SYS58451).

Un'altra modifica, che può tornare utile quando si vogliano cancellare da un programma molte righe consecutive, è la seguente: POKE774,0. Se adesso listate il gruppo di righe da eliminare, appariranno i SOLI numeri di riga: andate col cursore sul primo di essi, premete più volte il tasto Return... e il gioco è fatto! Per ripristinare, eseguite: Poke 774,26.

Il programma pubblicato

"Remkiller" è un breve programma in l.m. che fornisce un esempio di come si possa inserire un "wedge" nel comando LIST modificando il puntatore 774-775.

Caricate il programma in memoria, e il puntatore verrà automaticamente deviato ad una routine che parte da $02ED e che controlla se il carattere che sta per essere stampato è una REM (rappresentata in Basic dal valore 143), saltando, in caso affermativo, direttamente alla riga successiva. Ciò significa che tutte le REM, e i caratteri che le seguono fino alla fine della riga, scompariranno dal vostro listato sullo schermo, rendendo di fatto inefficace qualunque "protezione anti-LIST" basata sui REM con caratteri speciali.

```
10 REM REM KILLER
20 :
30 REM BY ROBERTO MORASSI
40 REM -PISTOIA-
45 :
50 REM PROVATE A LASCIARE QUES
   TE
60 REM REM PRIMA DI LANCIARE I
   L
70 REM BREVISSIMO PROGRAMMA
80 REM CHE SERVE PER SPROTEGGE
   RE
90 REM ALCUNI LISTATI PROTETTI
100 REM RICORRENDO AI CARATTERI
110 REM SPECIALI DOPO LE REM
120 :
130 DATA 8,201,143,208,10,36,15
   ,48,6,40,160,0,76,8,167,40,
   76,26,167,139,227
140 DATA 131,164,124,165,237,2:
   REM LASCIATE QUESTA REM...
150 FOR X=0 TO 26:READ Y:POKE 7
   49+X,Y:NEXT:REM ...E ANCHE
   QUESTA
```

COMPRAMI E TI FARO' UN REGALO!

Acquistando due scatole di Flexible Disk MEMOREX puoi chiedere subito un omaggio simpatico, originale e utile:

l'orologio impermeabile sport-time MEMOREX con il portamonete da polso.

è importante scegli
MEMOREX
A Burroughs Company

Giocate gente, ma... meditate!

Come evitar di perder denaro al gioco, come rischiare poco e bene ed altre cose ancora

di Alessandro de Simone

Il primo listato che proponiamo risolve uno dei principali problemi che si presentano in molti giochi di carte:

Problema: supponendo che un mazzo di carte sia composto da un numero eguale di carte di Cuori, Quadri, Fiori, Picche, quante e quali sono le possibili combinazioni che possono verificarsi estraendo a caso quattro carte dal mazzo?

Se, in altre parole, da un mazzo di carte prendiamo quattro carte, ecco alcune delle combinazioni che possono capitare:

cuori, picche, cuori, fiori
picche, picche, fiori, picche
fiori, cuori, cuori, picche

Come si può fare, dunque, per calcolare quante sono, in totale, le diverse possibilità?

Il calcolo da effettuare in questo caso, e in tanti altri che hanno la stessa "matrice" richiede l'esecuzione di una potenza.

Per risolvere il problema è necessario effettuare il seguente calcolo:
N.combinazioni= numero dei semi elevato al numero di estrazioni.

Ne deriva che:
N.combin=4#4=256

Se, ad esempio, togliete dal mazzo tutte le carte di cuori ed estraete, in seguito, ancora quattro carte, le possibilità sono:
N.com=3#4=81

Lo stesso concetto viene utilizzato anche in informatica nel risolvere un classico problema:
Problema: tenendo conto che in un byte (=8 bit = 8 posizioni = 8 "estrazioni") ciascun bit può assumere solo il valore "1" (=cuori) oppure "0" (=fiori), quante possibili combinazioni di "1" e "0" possono verificarsi?
Soluzione:
N.com.=2#8=256

Il primo listato

Il mini listato pubblicato ("Prima esperienza") consente di visualizzare le 256 combinazioni possibili del primo problema proposto.

Per natura, noi della Systems, siamo pacifici, prudenti e (ragionevolmente) razionali.

Con un senso di fastidio, quindi, leggiamo che c'è gente che, a cuor leggero, sperpera centinaia di milioni sui tavoli verdi dei casinò affidando ingenti somme al caso, alla fortuna oppure, più semplicemente, all'incoscienza.

Più che fastidio, però, proviamo rabbia quando sui periodici specializzati ci capita di leggere inserzioni pubblicitarie che assicurano (dietro congruo compenso) vincite clamorose, proponendo sistemi "sicuri" per realizzare elevati punteggi al totocalcio, lotto, totip ed altri giochi similari.

Inutile dire che se i sistemi proposti necessitano di computer (nella fattispecie, Commodore 64), ci chiediamo in che misura un utente di calcolatori può credere alle promesse pubblicate.

Da questo numero di Commodore Computer Club, pertanto, inizia una nuova "sezione" dedicata ai giochi d'azzardo.

E ciò non viene certo deciso per illudere i nostri lettori o, peggio, per "iniziarli" a pratiche sciagurate come il riversare mezzo stipendio al bancolotto.

Nostra intenzione, al contrario, è di convincere la gente che non è possibile escogitare un sistema "sicuro" o, comunque, tale che garantisca una vincita matematicamente superiore alla spesa sostenuta per partecipare al gioco.

La parte del leone, ovviamente, la prenderanno i listati che prendono in esame le combinazioni, permutazioni, il calcolo delle probabilità, la valutazione del rischio e tutta quella miriade di argomentazioni, insomma, che affrontando il problema in modo (finalmente) scientifico, da una parte spingano il lettore a prender coscienza delle reali dimensioni del problema e, dall'altra, a riconoscere come irragionevole la pratica suicida di incrementare sempre più le somme da destinare al gioco in una rincorsa senza fine.

E se qualche programma pubblicato sarà da voi ritenuto sufficientemente attendibile e degno di verifica, giocate pure qualche colonna al totocalcio o un terno al lotto, ma senza uscire dai limiti della decenza.

In caso di vincita, ovviamente, comunicatecelo, magari inviando in Redazione una bottiglia di spumante.

Partecipazione dei lettori

I lettori sono invitati a partecipare inviando programmi, corredati di articoli, entrambi scritti su supporto magnetico, nastro o disco, utilizzando uno dei seguenti Word Processor: Easy Script; Word Pro III; Word Pro del Plus/4; Superscript/128; Magic Desk; Wordcraft Vic/20.

Gli argomenti affrontati dovranno esser relativi ai temi tipici dei giochi d'azzardo: calcolo di combinazioni, calcolo delle probabilità, determinazione della percentuale di vittoria tra due squadre di calcio, eccetera.

Allo scopo di evitare l'invio di materiale non idoneo alla pubblicazione, gli aspiranti collaboratori sono pregati di contattare telefonicamente la Redazione, solo al pomeriggio (tel.02/8467348) chiedendo dell'Ing. de Simone.

Il suo funzionamento è banale: dopo aver assegnato ai primi quattro elementi del vettore X$() i nomi, rispettivamente, di "Cuori", "Picche", "Quadri" e "Fiori", si fa ricorso a quattro cicli For...Next nidificati l'uno all'interno dell'altro.

Il risultato consiste nella visualizzazione, su schermo, delle 256 combinazioni possibili precedute dal numero d'ordine scritto in reverse.

La visualizzazione è piuttosto rapida e non consente, in verità, un'agevole lettura. Per rallentare lo scrolling (scorrimento del video) sarà sufficiente tener premuto il tasto ⟨Ctrl⟩ (caso del Vic/20 e C/64) oppure il tastino Commodore (in basso a sinistra) nel caso si possegga il C/16 o Plus/4. Chi possiede il C/128 può utilizzare il tasto ⟨No Scroll⟩.

E' possibile, comunque, inserire la seguente linea che ha il compito di introdurre una pausa dopo ogni visualizzazione:

155 FOR I=1 TO 1000: NEXT

Totocalcio: primo programma

Le stesse considerazioni di prima vengono applicate nel programma "Combinazioni su 9 triple" che si propone di calcolare, e visualizzare (ciascuna in orizzontale, per motivi di praticità), tutte le combinazioni possibili nel caso di una giocata al totocalcio con nove triple. Il motivo di limitarsi a nove triple, invece che alle 13 che possono verificarsi in realtà, è dovuto soltanto alle limitazioni del computer: non è infatti possibile nidificare più di nove cicli For...Next (10 col C/16) se all'interno del "nocciolo" vi sono espressioni (righe 250-260, appunto) contenenti variabili, costanti oppure stringhe.

Il calcolo delle combinazioni possibili può comunque essere agevolmente effettuato:
N.comb.13 triple:
3*13=1.594.323 (!)

Il numero delle colonne possibili con nove triple "soltanto" (come proposto nel programma) risulta:
3*9=19.683

Ecco giustificata, quindi, la presenza delle righe 150 e 280 incaricate di determinare il tempo necessario al computer per calcolare, e visualizzare, le quasi ventimila colonne di nove simboli "1", "2" oppure "X".

Un altro problema

Torniamo, ora, al programma di prima, relativo all'estrazione di quattro carte, e facciamo alcune considerazioni:

Abbiamo detto, e verificato, che le combinazioni possibili, nel caso particolare esaminato, sono 256.

Un altro problema che si può risolvere, riferendoci ancora all'estrazione di quattro carte, può esser quello di determinare il numero di possibili combinazioni diverse tra loro che tengano conto del numero dei semi estratti, indipendendentemente dall'ordine di estrazione.

In parole più semplici: tra le 256 combinazioni possibili vi sono, tra le altre, le seguenti:

cuori, cuori, fiori, picche
picche, cuori, fiori, cuori
fiori, cuori, cuori, picche

che, in effetti, sono diverse tra loro se si considera l'ordine di estrazione.

Se però consideriamo il numero dei semi presenti in ciascuna estrazione, le tre appena esaminate risultano eguali tra loro dal momento che vi sono due cuori, un fiori ed un picche in ognuna di esse.

Come calcolare, dunque, il numero di estrazioni diverse tra loro tenendo conto solo del numero di semi presenti?

Il terzo programma

Questo programma ("Seconda esperienza") risolve il problema accennato attraverso un procedimento suscettibile di miglioramenti. Nell'invitare il lettore a studiare il listato pubblicato, ci limitiamo ad accennare il problema e la soluzione adottata.

All'interno della matrice bidimensionale Z(256,4) vengono depositate, a mano a mano che sono elaborate, le singole "estrazioni". Nella posizione "0", di ciascun elemento, sarà presente il numero di cuori estratti; analogamente, nelle successive posizioni "1", "2" e "3" di ciascun rigo della matrice, saranno presenti il numero, variabile tra zero e quattro, della presenza degli altri semi.

Supponiamo, ad esempio, le quattro estrazioni successive:

cuori, cuori, picche, quadri
cuori, cuori, picche, fiori
cuori, cuori, quadri, cuori
cuori, cuori, quadri, picche

Il contenuto delle quattro righe di matrice corrispondenti saranno, di conseguenza:

2,1,1,0
2,1,0,1
3,0,1,0
2,1,1,0

nell'ipotesi, ovviamente, che la prima posizione sia relativa ai cuori, la seconda alle picche, la terza ai quadri e l'ultima ai fiori.

Si può notare subito, nell'esempio riportato, che la prima e l'ultima riga, benchè diverse come ordine di estrazione, risultano eguali per contenuto (due cuori, un picche un quadri).

Ad individuare tutte le righe eguali, per accettare cioè soltanto la prima di esse e scartare tutte le altre, provvede la parte del programma compresa tra 290 e 400. Si comincia confrontando il contenuto della prima riga della matrice con tutte le righe successive: non appena viene individuato un "doppione", il programma (riga 390), provvede ad apporre un "1" nella quinta posizione della stessa riga della matrice. Tale simbolo (1, appunto), servirà in seguito per evitare confronti superflui che farebbero solo perder tempo.

Tanto per esser più chiari, la parte di matrice prima vista diventa quindi:

2,1,1,0,0
2,1,0,1,0
3,0,1,0,0
2,1,1,0,1

Terminato il confronto della prima riga di matrice con tutte le altre, un

paragone sarà quindi realizzato tra la seconda riga e le successive; poi tra la terza e le successive e così via.

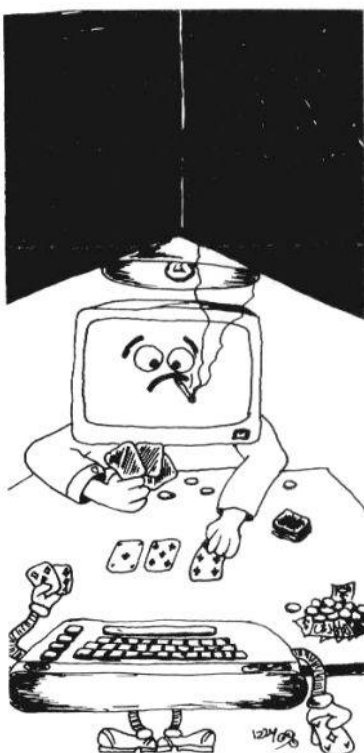
Poichè il confronto (righe 360-380) richiede tempo, ecco spiegata l'utilità del codice "1" in quinta posizione. Prima di effettuare il confronto, infatti, si esamina il codice della riga di matrice: se è presente un "1" è inutile effettuare il confronto dal momento che la riga di matrice risulta già "scartata" da un confronto avvenuto in precedenza. Si noti che, ovviamente, si evita di fare un intero ciclo (If...Then di riga 310) nel caso in cui la quinta posizione contenga, appunto, un "1".

Che cosa fa il programma

Non appena si impartisce il Run, compare il messaggio "Combinazione N." seguita da un valore, continuamente aggiornato, che parte da zero per arrivare a 255 (per un totale, appunto, di 256).

Al di sotto, in rapida successione, sono visualizzate, una alla volta, le 256 combinazioni possibili. In questa fase di elaborazione (righe 150-280) viene effettuata la somma dei semi presenti in ciascuna estrazione e depositata in ciascun elemento delle 256 righe della matrice. Si ricorda che, al momento del Run, il quinto elemento di ciascuna riga di matrice (quello numerato con... 4) è sempre nullo.

Subito dopo viene visualizzato il messaggio "Combinazione in esame" seguito dal numero della riga della matrice elaborata in quel momento. In alto a sinistra compare il numero delle righe esaminate in rapida successione. Non appena viene incontrata una riga eguale a quella in esame, viene posto un "1" nella quinta posizione e, contemporaneamente, compare il messaggio "Prossimo scarto" seguito, appunto, dal numero di riga che, in seguito, non parteciperà più al confronto, velocizzando, di conseguenza, le operazioni.

Dopo un certo numero di combinazioni in esame comparirà anche il messaggio "Combinazione scartata" seguita dal numero di riga che, contenendo un "1" in quinta posizione,

Poichè desideriamo che chiunque (anche gli studenti delle medie) sia in grado di comprendere queste note, ci permettiamo di ricordare il concetto di "potenza" scusandoci con gli esperti di matematica.

Per "potenza" di un numero, dunque, si intende il prodotto dello stesso numero, moltiplicato per se stesso, per un certo numero di volte. Ad esempio:
10 elevato alla potenza di 3 significa:
10*10*10=1000
7 elevato alla potenza di 4:
7*7*7*7=2401
eccetera.

Il simbolo di "elevato a" si indica con una freccia in alto. Esempio:
3 elevato alla 3 si scrive: 3#3=27

I computer Commodore utilizzano proprio questa simbologia. Provate, ad esempio:
PRINT 4#5
o altre similari.

L'eventuale messaggio "Overflow Error" indica che il numero da calcolare è troppo grande.

non è stato presa in considerazione.

Il tempo totale di elaborazione è di poco superiore ai nove minuti e le combinazioni diverse, alla fine, risultano 34 sulle 256 di partenza.

L'ultimo programma

Il listato "Sviluppo sistema" deve esser considerato come base di partenza per l'attento lettore.

Si riferisce, infatti, a cinque combinazioni (da incrementare o diminuire con semplici modifiche) di cui due triple e tre doppie.

Per ciò che riguarda lo sviluppo di una tripla non vi sono particolari problemi dal momento che devono esser setacciate tutte le combinazioni possibili.

Nel caso delle doppie, al contrario, è indispensabile riferirsi ad una sola delle tre combinazioni da impostare:

* 1/2
* 1/X
* X/2

Nel caso di sviluppo di un sistema totocalcio, infatti, la doppia 1/2, ad esempio, genera gli stessi risultati se considerata come 2/1.

Per evitare complicate elaborazioni si è preferito assegnare due volte il simbolo "X" (riga 140):

A$(0)="X":A$(3)="X"

In seguito si agisce sui cicli For...Next per selezionare, in caso di doppie, quella scelta tra le tre possibili (righe 170-190)

Conclusioni

I listati pubblicati non sono altro che un assaggio di ciò che, in futuro, vedrete su Commodore Computer Club.

In attesa di vincere milioni, comunque, impegnatevi a sofisticare i programmi di queste pagine introducendo altre opzioi come la memorizzazione su disco o nastro dei risultati, la stampa su carta, lo sviluppo di altre ipotesi.

Tali operazioni, se non altro, sono gratis e non c'è nulla da rischiare...

Primo programma

```
100 REM  I GIOCHI D'AZZARDO
110 REM  LE COMBINAZIONI POSSIB
    ILI
120 REM  PRIMA ESPERIENZA
130 :
140 X$(0)=" CUORI":X$(1)=" PICC
    HE":X$(2)=" QUADRI":X$(3)="
     FIORI"
150 FOR A=0 TO 3:FOR B=0 TO 3:F
    OR C=0 TO 3:FOR D=0 TO 3
160 PRINTCHR$(18)WCHR$(146);:W=
    W+1
170 PRINTX$(A)X$(B)X$(C)X$(D):N
    EXTD,C,B,A
```

Secondo programma

```
100 REM  I GIOCHI D'AZZARDO
110 REM  IDEA BASE PER TOTOCALC
    IO:
120 REM  SVILUPPO SISTEMA
130 :
140 A$(0)="X":A$(1)="1":A$(2)="
    2":A$(3)="X":W=1
150 FOR A=0 TO 2:REM  TRIPLA
160 FOR B=0 TO 2:REM  TRIPLA
170 FOR C=1 TO 2:REM  DOPPIA (1
    /2)
180 FOR D=0 TO 1:REM  DOPPIA (1
    /X)
190 FOR E=2 TO 3:REM  DOPPIA (X
    /2)
195 PRINTA$(A)A$(B)A$(C)A$(D)A$
    (E);
200 PRINTCHR$(18)W:W=W+1
210 NEXTE,D,C,B,A
```

Terzo programma

```
100 REM  I GIOCHI D'AZZARDO
110 REM  LE COMBINAZIONI POSSIB
    ILI
120 REM  SECONDA ESPERIENZA:
130 REM  ESCLUSIONE DEI CASI EG
    UALI
140 :
150 DIM Z(255,4):REM  N.COMBIN.
    POSSIBILI
160 X$(0)=" CUORI":X$(1)=" PICC
    HE":X$(2)=" QUADRI":X$(3)="
     FIORI":PU=0
170 PRINTCHR$(147)"COMBINAZIONE
     N."
180 FOR A=0 TO 3:FOR B=0 TO 3:F
    OR C=0 TO 3:FOR D=0 TO 3
190 PRINTCHR$(19) TAB(16)CHR$(1
    8)WCHR$(146):W=W+1
200 PRINT:PRINTX$(A)X$(B)X$(C)X
    $(D)"     "
210 FOR T=0 TO 3:X(T)=0:NEXT:FO
    R T=0 TO 3
220 IF X$(T)=X$(A) THEN X(T)=X(
    T)+1
230 IF X$(T)=X$(B) THEN X(T)=X(
    T)+1
240 IF X$(T)=X$(C) THEN X(T)=X(
    T)+1
250 IF X$(T)=X$(D) THEN X(T)=X(
    T)+1
260 NEXT:REM  FORT=0TO3:PRINTX(
    T);:NEXT:PRINT
270 FOR T=0 TO 3:Z(PU,T)=X(T):N
    EXT:PU=PU+1
280 NEXTD,C,B,A
290 REM ****CONFRONTO TRA PARTI
     SIMILI**
300 TI$="000000"
310 FOR W=0 TO 254:PRINTCHR$(14
    7)"     COM.IN ESAME:"W:IF Z
    (W,4) THEN 410
320 R=0:FOR Y=0 TO 3:IF Z(W,Y)=
    4 THEN R=1
330 NEXT:IF R THEN 410
340 FOR X=W+1 TO 254:U=0:PRINTC
    HR$(19)X
350 IF Z(X,4)=1 THEN PRINTCHR$(
    19)CHR$(17)"COMB.SCARTATA:"
    X:GOTO 400
360 FOR Y=0 TO 3
370 IF Z(W,Y)=Z(X,Y) THEN U=U+1
380 NEXTY
```

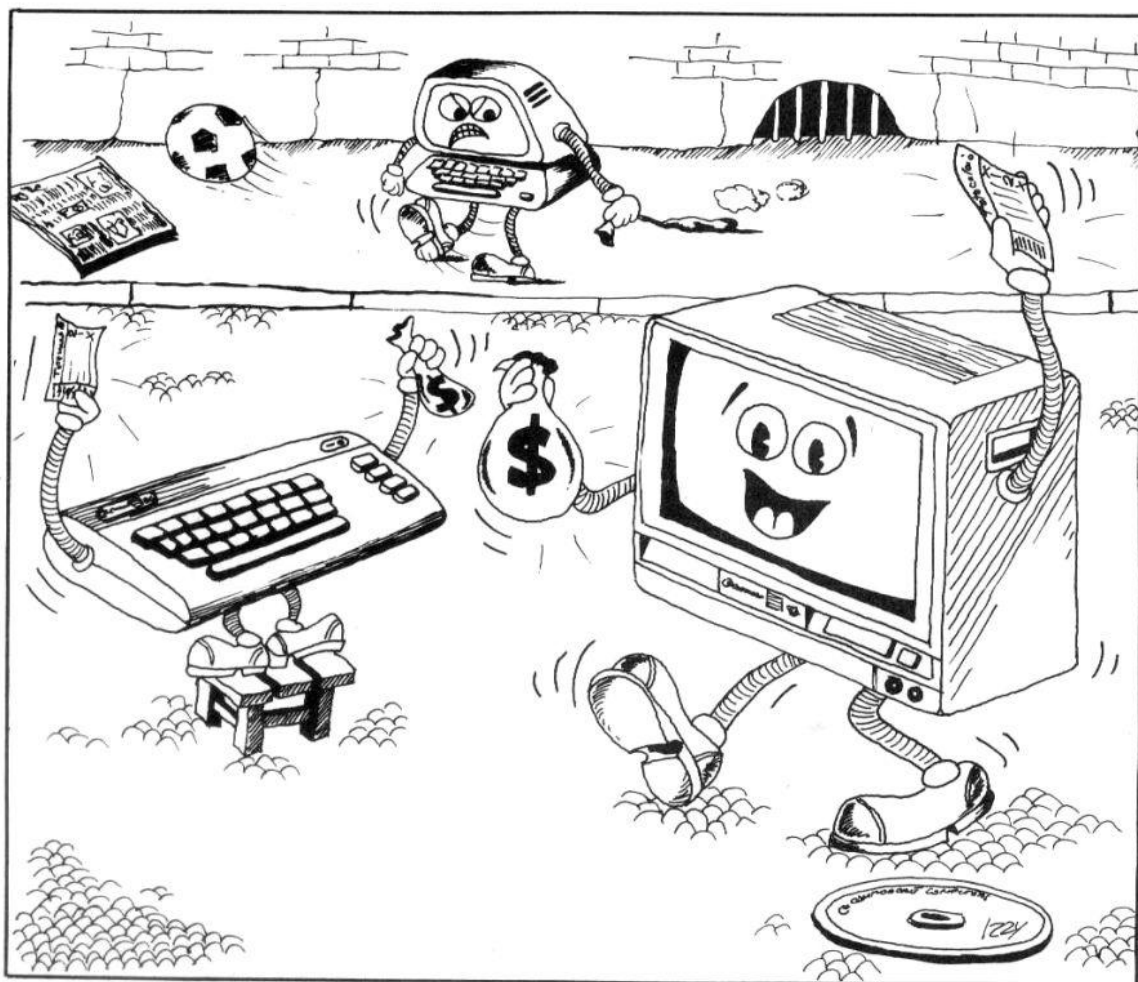
```
390 IF U=4 THEN PRINTCHR$(19)CH
    R$(17)CHR$(17)"PROSSIMO SCA
    RTO:"X:Z(X,4)=1
400 NEXTX
410 NEXTW:PRINT"TEMPO TRASCORSO
    :"TI$:REM  9 MIN 6 SECONDI
420 PRINT"PER ESAMINARE TUTTE L
    E POSSIBILI COMBINAZIONI PR
    EMI <RETURN>"
430 GET A$:IF A$="" THEN 430
440 IF ASC(A$)<>13 THEN 420
450 X=0:FOR PU=0 TO 255:IF Z(PU
    ,4) THEN 470
460 PRINTCHR$(18)PU;:FOR T=0 TO
     3:PRINTCHR$(146)Z(PU,T);:N
    EXTT:PRINT:X=X+1
470 NEXTPU:PRINT:PRINT"N.TOTALE
     COMBINAZIONI DIVERSE="X
480 END
```

Quarto programma

```
100 REM  TOTOCALCIO:
110 REM  N.TOTALE DI COMBINAZIO
    NI
120 REM  (SU 9 TRIPLE)
130 :
140 A$(0)="X":A$(1)="1":A$(2)="
    2":X=1
150 TI$="000000"
160 FOR A=0 TO 2:REM  PARTITA N
    .1
170 FOR B=0 TO 2
180 FOR C=0 TO 2
190 FOR D=0 TO 2
200 FOR E=0 TO 2
210 FOR F=0 TO 2
220 FOR G=0 TO 2
230 FOR H=0 TO 2
240 FOR I=0 TO 2:REM  PARTITA N
    .9
250 PRINTA$(A)A$(B)A$(C)A$(D)A$
    (E);
260 PRINTA$(F)A$(G)A$(H)A$(I)CH
    R$(18)X
270 X=X+1:NEXTI,H,G,F,E,D,C,B,A
280 PRINT"TEMPO TRASCORSO "TI$
```

Sviluppo di sistemi condizionati

Come estrarre da un "sistema" per Totocalcio, Totip, Enalotto e similari solo le colonne che hanno la massima probabilità di vittoria, escludendo tutte le altre

di Antonio Pastorelli

I sistemi condizionati sono quei sistemi basati su dati statistici.

Esaminando le colonne vincenti, nella storia del Totocalcio, si può notare che il numero di segni "1", "X" e "2", si aggira, salvo eclatanti eccezioni, intorno a intervalli piuttosto ristretti.

Ad esempio il numero di segni "2" presente in una colonna vincente è sempre limitato: raramente si realizzano più di quattro vittorie fuori casa.

Così come per il numero dei segni, si può osservare che non compaiono mai, salvo casi eccezionali, 6 segni "1" consecutivi come pure 3 segni "2" l'uno di seguito all'altro.

Il programma TOT 13, permette, partendo da un sistema-base predeterminato dall'utente, di eliminare, tra tutte le colonne possibili in teoria, quelle che non soddisfano determinate condizioni relative al numero minimo e massimo dei segni, e alla consecutività.

Riferendoci, ad esempio, ad un pronostico relativo a due incontri di calcio, il cui risultato è del tutto incerto, è indispensabile assegnare due triple al sistema base.

Nonostante i due incontri siano del tutto incerti, possiamo però ragionevolmente stabilire che non può verificarsi l'evento di due pareggi in contemporanea: siamo convinti che almeno una delle quattro squadre vinca la partita (=al massimo un solo pareggio).

Come fare, dunque, se si vuole prevedere un qualsiasi risultato, per i due incontri accennati, eliminando solo la possibilità che in entrambi si pareggi?

Il problema è risolto dal programma pubblicato in queste pagine grazie all'opzione delle "Colonne impossibili".

Questa permette di eliminare dallo sviluppo del sistema le colonne con determinati risultati indipendentemente dalle altre condizioni imposte. Nel caso esposto, nel sistema-base compariranno due triple, e in più vi sarà una colonna impossibile che contiene, in corrispondenza dei due incontri, due segni "X", che ricorderà al calcolatore di scartare le colonne contenute nel sistema che eventualmente prevedano il pareggio in entrambi gli incontri.

Nel digitare righe di programma basic che contengono istruzioni DATA, è piuttosto facile incorrere in errori di digitazione. Supponiamo che un'ipotetica linea basic numerata con 1200 debba contenere i tre valori: 123, 456, 789. Ecco alcuni esempi di errori più frequentemente commessi:

```
1200 DATA,123,456,789
```

C'è una virgola dopo la parola "DATA". I dati letti dal computer sono, in questo caso, quattro: 0, 123, 456, 789. Se, infatti, non figura alcun carattere dopo l'istruzione DATA, automaticamente viene assunto il valore nullo (0).

```
1200 DATA 123,456,789,
```

In questo caso, dopo il numero 789, il computer, grazie alla presenza della virgola erroneamente inserita, "crede" che ci sia un altro valore e, non trovandolo, lo assume come nullo (0).

```
1200 DATA 1234,56,789
```

La virgola è posizionata male, vale a dire dopo il carattere "4" e non dopo il carattere "3". Il computer non può sapere se il valore esatto è 123 oppure 1234 e individuare un errore, in questo caso, risulta piuttosto laborioso.

I programmi

Il programma, in realtà, è diviso in due parti: TOT 13 e DISPLAY.

TOT 13 provvede allo sviluppo vero e proprio del sistema e a memorizzare le colonne selezionate sui supporti di massa (nastro o disco). In seguito, con DISPAY, si visualizzeranno con comodo, su schermo, le colonne valide.

Tale soluzione è stata adottata perchè, in certi casi, il numero di colonne può esser rilevante. L'utilizzatore, dal momento che deve trascrivere a mano sulla schedina (purtroppo) ciascuna colonna elaborata, potrebbe decidere di effettuare la trascrizione in più riprese.

Come utilizzare Tot 13

Dopo averlo caricato e lanciato (e atteso qualche secondo), verrà chiesto il sistema-base, al quale bisogna rispondere tenendo presente che le varianti vanno introdotte mantenendo l'ordine di priorità dei segni (pri-

ma il segno "1", poi "X", quindi il "2"); ad esempio il computer accetta la variante "X2" e rifiuta "2X". Esempio di videata:

SISTEMA-BASE
1? 1
2? 2
3? 1
4? 1
5? 1X
6? X
7? X
8? 1X
9? X
10? 1
11? 1
12? X2
13? 1X2

Viene poi richiesto, per ognuno dei tre segni, il numero minimo e massimo di presenze accettate per ogni colonna elaborata. Esempio:

NUMERO SEGNI
1-DA? 5
1-A? 7
X-DA? 3
X-A? 5
2-DA? 1
2-A? 3

Successivamente bisogna indicare la consecutività massima (cioè quanti segni uguali di seguito, al massimo, possono presentarsi). Esempio:

NUMERO MAX SEGNI
CONSECUTIVI
1-CONS MAX? 3
X-CONS MAX? 5
2-CONS MAX? 1

A questo punto il computer chiede se si vogliono inserire colonne-impossibili: rispondendo negativamente (N) il programma inizierà a sviluppare il sistema; in caso contrario bisognerà inserire da una fino a cinque colonne impossibili. Se si sbaglia, si può rispondere alla prima con ⟨Return⟩ e poi con il tasto asterisco (*).

Alle colonne impossibili bisogna rispondere con 'return' in corrispondenza dei pronostici che non interessano, e mettendo il segno che non deve comparire nelle colonne valide, per gli altri.

Dopo aver inserito la prima colonna impossibile, premendo (*) inizia l'elaborazione, mentre premendo un qualsiasi altro tasto si passa a quella successiva (ripetiamo che se ne possono inserire al massimo 5).

Una nota per i possessori di C/16 e Plus/4: durante la fase di sviluppo l'immagine video scompare (come quando si usa il registratore); questo perchè il computer non dovendo gestire il video, avrà un risparmio in velocità di esecuzione del 33,5%

Il lavoro di selezione delle colonne valide risulta piuttosto veloce dal momento che è scritto in linguaggio macchina. A questo punto potete decidere se registrare le colonne valide (premendo "S" alla domanda "Registri?"), oppure di visulaizzarle subito in caso di risposta negativa. Potete, naturalmente, sviluppare di nuovo il sistema, cambiando i condizionamenti (per tentare di ridurre il numero eventualmente eccessivo di colonne), o di sviluppare un altro sistema.

E' importante sottolineare che il file sequenziale automaticamente generato ha, come nome, "TOT" (vedi riga 740) eventualmente modificabile a piacere. Gli utilizzatori del disco non potranno, quindi, scrivere più di un file per ciascun disco a meno di non cambiare il nome del file stesso prima di far ripartire il programma.

Si è preferito, infatti, evitare il ricorso al comando della famigerata "chiocciolina" che, come è noto, può provocare guai nella directory.

Ricordiamo, inoltre, che alcune righe devono esser trascritte ricorrendo alle abbreviazioni dei comandi Basic per evitare di "uscire" dai limiti degli 80 caratteri consentiti.

Come si utilizza Display

Con DISPLAY si caricano in memoria le colonne precedentemente registrate, in modo da visualizzarle sullo schermo.

Nel copiare le colonne sulle schedine, vi aiuterà la freccetta posta sotto di esse, controllabile mediante i tasti-cursore.

Il programma gira su: VIC-20+esp. min 8K, C-16, C-64, Plus/4, C-128.

Note per gli esperti:

Il file che contiene le colonne vincenti è formattato nel modo seguente: contiene come primo dato il numero delle stesse, seguito dal blocco consecutivo di tutte le colonne elaborate.

All'interno dello stesso file, il numero 1 corrisponde al simbolo "1", il 2 al pareggio (X) ed il 3 alla vittoria fuori casa (2).

Avvertenze

E' facile "farsi prendere la mano" giocando al Totocalcio. Ricordiamo, pertanto, che non è assolutamente possibile escogitare un sistema "certo": a vincere è sempre e solo l'organizzatore (leggi CONI e Stato). Sconsigliamo vivamente, quindi, di dedicare grosse cifre a quello che dovrebbe esser considerato solo un gioco e che, purtroppo, viene invece reclamizzato dai mass media, spesso anche in modo subliminale, incoraggiando la gente a rischiare somme consistenti.

```
100 REM TOT-13
110 REM PER QUALSIASI COMMODORE
120 REM BY ANTONIO PASTORELLI
125 :
130 C$=CHR$(147):D$=CHR$(145):E
    $=CHR$(18):F$=CHR$(146)
140 PRINTC$"TOT 13":J1=9499:J2=
    9512:POKE 55,28:POKE 56,37
150 FOR J=1 TO 450:READ A:U=U+A
160 POKE 9849+J,A:NEXT:IF U<>51
    078 THEN PRINT"ERRORE NEI D
    ATA":END
170 DIM B$(13)
180 PRINTC$"SISTEMA-BASE":FOR J
    =1 TO 13:PRINT TAB(3-LEN(ST
    R$(J)));J;
190 INPUT A$:B$(J)=A$
200 IF A$="1" THEN POKE J1+J,1:
    POKE J2+J,1:GOTO 280
210 IF A$="X" THEN POKE J1+J,2:
    POKE J2+J,2:GOTO 280
220 IF A$="2" THEN POKE J1+J,3:
    POKE J2+J,3:GOTO 280
230 IF A$="1X" THEN POKE J1+J,4
    :D=D+1:POKE J2+J,1:GOTO 280
240 IF A$="12" THEN POKE J1+J,5
    :D=D+1:POKE J2+J,1:GOTO 280
250 IF A$="X2" THEN POKE J1+J,6
    :D=D+1:POKE J2+J,2:GOTO 280
```

```
260 IF A$="1X2" THEN POKE J1+J,
    7:TR=TR+1:POKE J2+J,1:GOTO
    280
270 GOTO 180
280 NEXT
290 POKE 3,104:POKE 4,37:POKE 5
    ,61:POKE 6,40
300 J=INT(2↑D*3↑TR):J1=J:D=J/65
    281:POKE 9528,INT(D):J=J-IN
    T(INT(D)*65281)
310 D=J/256:POKE 9527,INT(D):J=
    J-INT(INT(D)*256)
320 POKE 9526,J
330 FOR J=1 TO 13
340 IF LEN(B$(J))=1 THEN POKE 9
    528+J,1:POKE 9541+J,0:POKE
    9554+J,0
350 IF LEN(B$(J))=2 THEN J1=J1/
    2:GOSUB 380
360 IF LEN(B$(J))=3 THEN J1=J1/
    3:GOSUB 380
370 NEXTJ:GOTO 400
380 J2=J1:D=J1/65281:POKE 9554+
    J,INT(D):J1=J1-INT(INT(D)*6
    5281)
390 D=J1/256:POKE 9541+J,INT(D)
    :J1=J1-INT(INT(D)*256):POKE
     9528+J,J1:J1=J2:RETURN
400 PRINTC$"NUMERO SEGNI"
410 INPUT "1-DA";D:GOSUB 1100:P
    OKE 9682,D:INPUT "1-A";A:GO
    SUB 1120:POKE 9570,A:A=0:D=
    0
420 INPUT "X-DA";D:GOSUB 1100:P
    OKE 9683,D:INPUT "X-A";A:GO
    SUB 1120:POKE 9571,A:A=0:D=
    0
430 INPUT "2-DA";D:GOSUB 1100:P
    OKE 9684,D:INPUT "2-A";A:GO
    SUB 1120:POKE 9572,A:D=0
440 PRINTC$"NUMERO MAX SEGNI":P
    RINT"CONSECUTIVI"
450 INPUT "1-CONS MAX";D:GOSUB
    1140:POKE 9573,D:D=0
460 INPUT "X-CONS MAX";D:GOSUB
    1140:POKE 9574,D
470 D=0:INPUT "2-CONS MAX";D:PO
    KE 9575,D
480 PRINTC$"INSERISCI":PRINT"CO
    LONNE IMPOSSIBILI?"
490 GET A$:IF A$="S" THEN 520
500 IF A$="N" THEN FOR J=9576 T
    O 9640:POKE J,0:NEXT:GOTO 6
    40
510 GOTO 490
520 FOR X=1 TO 5:PRINTC$"COLONN
    A IMPOSS. N";X:PRINT"*=END"
530 A=9562:FOR J=1 TO 13:B$=""
540 PRINT TAB(3-LEN(STR$(J)));J
    ;:INPUT B$
550 IF B$="" THEN POKE A+X*13+J
    ,0:GOTO 600
560 IF B$="1" THEN POKE A+X*13+
    J,1:GOTO 600
570 IF B$="X" THEN POKE A+X*13+
    J,2:GOTO 600
580 IF B$="2" THEN POKE A+X*13+
    J,3:GOTO 600
590 PRINTD$;:B$="":GOTO 540
600 NEXTJ
610 GET A$:IF A$="*" THEN FOR J
    1=X+1 TO 5:FOR J=1 TO 13:PO
    KE A+J1*13+J,0:NEXTJ,J1:GOT
    O 640
620 IF A$="" THEN 610
630 NEXT
640 PRINTC$:FOR J=9641 TO 9681:
    POKE J,0:NEXT
650 FOR J=9685 TO 9695:POKE J,0
    :NEXT:CLR :SYS9850
660 IF PEEK(44)=16 THEN POKE 65
    286,PEEK(65286) OR 16
670 PRINTC$"COLONNE:";:A=PEEK(9
    641)+256*PEEK(9642):PRINTA
680 PRINT:PRINT"REGISTRI?"
690 GET A$:IF A$="S" THEN 720
700 IF A$="N" THEN 1160
710 GOTO 690
720 PRINTC$"NASTRO/DISCO (N/D)"
730 GET A$:IF A$="N" THEN OPEN
    1,1,1,"TOT":GOTO 760
740 IF A$="D" THEN OPEN 1,8,12,
    "TOT,S,W":GOTO 760
750 GOTO 730
760 PRINT#1,A:FOR J=10301 TO 10
    301+13*A-1
770 PRINT#1,PEEK(J):NEXT:CLOSE
780 DATA 165,44,201,16,208,8,17
    3,6,255,41,239,141,6,255,16
    0,13,185,40,37,170,254
790 DATA 212,37,173,216,37,236,
    216,37,208
800 DATA 19,254,216,37,189,216,
    37,221,219,37,144
810 DATA 13,240,11,157,219,37,7
    6,177,38,169,1,157,216,37,1
    42,216,37,136,208,211
820 DATA 160,3,185,212,37,217,2
    09,37,176,5
830 DATA 240,3,76,70,39,217,97,
    37,240,5,144,3,76
840 DATA 70,39,136,208,230,160,
    3,185,219,37
850 DATA 217,100,37,240,5,144,3
    ,76,70,39,76,26
860 DATA 40,234,177,3,201,0,240
    ,10,217,41
870 DATA 37,208,8,169,1,141,223
    ,37,76,42,40,169,0
880 DATA 76,53,40,173,223,37,20
    1,1,208,3,76
890 DATA 70,39,24,165,3,105,13,
    133,3,144,2,230
900 DATA 4,165,3,201,169,208,9,
    165,4,201,37
910 DATA 208,3,76,42,39,169,0,1
    41,223,37,76,35
920 DATA 40,162,1,160,0,189,40,
    37,145,5,230,5,208,2,230,6,
    232,224,14,208,238,238
930 DATA 169,37,208,3,238,170,3
    7,169,104,133,3,169,37,133,
    4,206,54,37,173,54,37
940 DATA 201,255,208,13,206,55,
    37,173,55,37,201,255,208,3,
    206,56,37,162,3,189,53
950 DATA 37,201,0,208,4,202,208
    ,246,96,162
960 DATA 11,169,0,157,212,37,20
    2,208,248,162,13
970 DATA 254,170,37,208,8,254,1
    83,37,208,3,254,196,37,202,
    208,240,162,13,189,170
980 DATA 37,221,56,37,208,19,18
    9,183,37,221
990 DATA 69,37,208,11,189,196,3
    7,221,82,37,208
1000 DATA 3,32,12,40,202,208,226
    ,76,136,38
1010 DATA 189,27,37,201,4,240,18
    ,201,5,240,32,201
1020 DATA 6,240,45,201,7,240,51,
    169,0,157,170
1030 DATA 37,96,189,40,37,201,1,
    208,6,169,2,157
1040 DATA 40,37,96,169,1,76,211,
    39,189,40,37,201,1,208,5,16
    9,3,76,211,39,169,1,76
1050 DATA 211,39,189,40,37,201,2
    ,208,221,76,227,39,189,40,3
    7,201,1,240,211,201,2
1060 DATA 240,5,169,1,76,211,39,
    169,3,76,211
1070 DATA 39,169,0,157,170,37,15
    7,183,37,157,196
1080 DATA 37,76,177,39,136,208,3
    ,76,35,40,76,213,38,162,0,1
    60,0,76,230,38,200,192
1090 DATA 13,208,3,76,254,38,76,
    230,38,141,223,37,76,254,38
    ,0
1100 IF D<0 OR D>13 THEN 400
1110 RETURN
1120 IF A<D  OR  A>13 THEN 400
1130 RETURN
1140 IF D<1 THEN 440
1150 RETURN
1160 PRINTC$;:FOR J=1 TO A:PRINT
    E$;J;F$;:FOR X=1 TO 13
1170 W=PEEK(10287+X+J*13)
1180 IF W=1 THEN PRINT"1";
1190 IF W=2 THEN PRINT"X";
1200 IF W=3 THEN PRINT"2";
1210 Q=Q+1:IF Q=3 THEN PRINT" ";
    :Q=0
1220 NEXT:PRINT:Q=0
1230 GET A$:IF A$="" THEN 1230
1240 NEXTJ
```

```
100 REM DISPLAY PER TOT 13
110 :
120 DIM A$(700):C$=CHR$(147):D$
    =CHR$(17):E$=CHR$(157):F$=C
    HR$(29)
130 FOR I=1 TO 17:G$=G$+CHR$(17
    ):NEXT:H$=CHR$(19)
140 PRINTC$"D I S P L A Y"
150 PRINT:PRINT"I DATI SONO SU
    NASTRO O DISCO? (N/D)"
160 GET A$:IF A$="N" THEN OPEN
    1,1,0,"TOT":GOTO 190
170 IF A$="D" THEN OPEN 1,8,12,
    "TOT,S,R":GOTO 190
180 GOTO 160
190 INPUT#1,NR
200 PRINTC$"LETTURA DATI"
210 FOR J=1 TO NR:FOR K=1 TO 13
    :INPUT#1,A$:A$(J)=A$(J)+A$:
    NEXTK,J:CLOSE 1
220 PRINTC$"OK"
230 PRINT"PREMI UN TASTO PER IN
    IZIARE"
240 GET A$:IF A$="" THEN 240
250 H=1
260 PRINTC$"DA";H;"A";H+19:P=0
270 PRINT:FOR J=H TO H+19
280 FOR K=1 TO 25 STEP 2:A$=MID
    $(A$(J),K,1)
290 IF A$="1" THEN 320
300 IF A$="2" THEN A$="X"
310 IF A$="3" THEN A$="2"
320 PRINTA$;D$;E$;:NEXTK
330 PRINTH$:PRINT:PRINT TAB(J);
340 NEXTJ:PRINTH$;G$"*:PROSSIMO
     GRUPPO":PRINT"[3 UP]↑";
350 GET Z$:IF Z$<>"*" THEN 410
360 IF Z$="" THEN 350
370 H=H+20
380 IF H<=NR THEN 260
390 PRINT:PRINT:PRINT:PRINT"FIN
    E"
400 END
410 IF Z$=CHR$(29) THEN P=P+1:G
    OTO 440
420 IF Z$=CHR$(157) THEN P=P-1:
    GOTO 460
430 GOTO 350
440 IF P<=19 THEN PRINT"[LEFT]
    ↑";:GOTO 350
450 P=19:GOTO 350
460 IF P>=0 THEN PRINT"[LEFT] [
    2 LEFT]↑";:GOTO 350
470 P=0:GOTO 350

    1:END
```

La macchina del tempo

Tutto quello che dovete conoscere sui due orologi interni del C/64

di Fabio Cestari

Forse non molti di voi sanno che nel C/64 esistono ben due orologi situati all'interno dei due CIA (Complex Interface Adapter), circuiti integrati che si occupano delle operazioni di Input/Output (I/O).

I registri dei due CIA partono rispettivamente da 56320 ($DC00) e da 56576 ($DD00).

Nel programmaa presentato in queste pagine si è fatto uso di uno solo dei due orologi; tuttavia il loro funzionamento è identico ed è sufficiente modificare l'indirizzo di partenza dei registri secondo i valori sopra riportati a seconda che si voglia utilizzare il primo o il secondo.

Il breve listato pubblicato è soltanto un esempio di questa tecnica di programmazione, che non sfrutta completamente le possibilità offerte dal computer.

Lasciamo a voi il piacere di modificarlo, completarlo e renderlo più interessante aggiungendo magari un cronometro ed un allarme.

A questo punto qualcuno di voi si sarà domandato se per caso l'ora di cui stiamo parlando non sia quella contenuta nell'arcinota variabile TI$.

Non lasciatevi ingannare da quest'ultima che non ha proprio nulla in comune con quella gestita dai due CIA.

Infatti la variabile TI$ è soltanto un

contatore incrementato ad ogni interrupt e per di più non è molto preciso considerato il fatto che quasi tutte le operazioni di I/O modificano la frequenza degli interrupt. Gli orologi di cui parliamo, invece, lavorano instancabilmente e la loro "avanzata" non viene minimamente alterata da "strane" operazioni che l'utente può compiere, come comandi di CLR, SYS64738, modifica involontaria di TI$ ed altre. Solo la pressione del tasto Reset (se lo possedete) azzera i due orologi.

L'ora

Passiamo a questo punto ad analizzare come è organizzata e come viene gestita l'ora all'interno del computer.

Facendo riferimento alla figura, potrete notare che esistono ben quattro registri che si occupano di conservare i dati per l'orario.

I dati ivi contenuti sono in forma BCD (Binary Coded Decimal) e ciò rende più semplice la conversione in caratteri ASCII (American Standard Code for Information Interchange).

La rappresentazione dei dati in forma BCD permette la codifica delle dieci cifre da "0" a "9" utilizzando soltanto 4 bit e quindi tutti i numeri superiori a 9 (1001 in codice binario) non sono numeri BCD legali; anche nel caso in cui due numeri BCD vengano sommati tra loro, e il risultato sia un numero maggiore di 9, tale valore non deve esser considerato in forma BCD ortodossa.

Passiamo ora ad analizzare uno per uno i contenuti dei registri rappresentati in figura.

Il registro 8 contiene i decimi di secondo rappresentati dai 4 bit meno significativi (dal bit 0 al bit 3) mentre i restanti 4 rimangono inutilizzati.

Nel 9 e nel 10 sono allocati, rispettivamente, i secondi e i minuti.

In entrambi, i primi 4 bit (meno significativi) contengono le unità, mentre gli altri 4 (più significativi) contengono le decine.

L'undicesimo registro contiene le ore e, come al solito, i primi 4 bit rappresentano le unità mentre dei restanti 4 il primo (bit n.4) contiene le decine (1 oppure 0) e l'ultimo (bit n.7) "1" se si è di pomeriggio (PM) e "0" se si è di mattina (AM).

Oltre a ciò è possibile fissare anche la sveglia (alarm).

Quest'ultima viene "scritta" negli stessi registri di cui abbiamo parlato precedentemente; tuttavia per fare ciò deve dapprima essere settato a 1 il bit 7 del registro 15 che va riportato a zero una volta terminata l'operazione di impostazione dell'allarme.

Per settare tale bit è sufficiente dare il comando:
POKE 56335,PEEK(56335) OR 128

Fate attenzione che il valore 56335 è stato ottenuto aggiungendo 15 al valore 56320 che è il valore di partenza dei registri del primo CIA; per il secondo CIA si dovrà aggiungere il valore 15 a 56576 che è appunto il valore di partenza del secondo CIA. In seguito faremo sempre riferimento al primo CIA.

Nel caso in cui si voglia aggiornare l'orario, si dovrà forzare a zero tale bit, cosa che si ottiene con il seguente comando:
POKE 56335,PEEK(56335) AND 127

Ricordate che questo comando deve essere dato anche ogni volta che termina un'operazione di aggiornamento dell'allarme.

Durante la fase di aggiornamento sia dell'ora che dell'allarme, è necessario seguire un certo ordine; infatti vanno aggiornate prima le ore poi i minuti, i secondi e per ultimi i decimi di secondo.

Non appena un dato viene posto nel registro delle ore, viene bloccato l'incremento dell'orario, che verrà ripristinato automaticamente subito dopo aver scritto nell'ultimo registro (quello dei decimi di secondo).

E' importante notare che mentre è sempre possibile sapere l'orario andando a leggere i valori contenuti nei quattro registri preposti a tale funzione, non lo è altrettanto per l'allarme.

Infatti pur settando a 1 il bit 7 del registro 15 e andando a leggere i registri dal n.8 all'undici, vi troveremo i valori delle ore, dei minuti e dei secondi dell'orario, mai quelli dell'allarme.

Per la conversione dell'ora in dati numerici in forma BCD tali, cioè, da poterli pokare nelle locazioni di cui abbiamo parlato, vi suggeriamo di dare un'occhiata alle linee del programma presentato, dalla 900 alla 940.

Il bit 7 del registro 14 svolge una particolare funzione di vitale importanza per la precisione dell'incremento dell'ora.

Tale bit serve per definire la frequenza di rete, frequenza che in altri Paesi è di 60 Hz. (bit 7 resettato) mentre in Italia è di 50 Hz (bit 7 a 1).

Poichè all'accensione del computer questo bit è automaticamente resettato, deve essere forzato a 1 affinchè il tutto possa funzionare con una certa precisione:
POKE 56334,PEEK(56334) OR 128

Cenni sulla gestione dell'allarme

Abbiamo già visto quale procedura si debba seguire per fissare l'allarme, tuttavia non abbiamo ancora parlato del suo funzionamento.

Una volta che sia stata fissata l'ora dell'allarme, il computer continuerà a confrontarla con quella in continuo aggiornamento, fino a che saranno uguali.

A questo punto viene settatto il bit 2 del registro 13 e attivato un interrupt.

Il registro 13, in verità, è formato da due registri che svolgono altrettante funzioni: il primo è a sola lettura e contiene tutte le possibili sorgenti d'interrupt, mentre il secondo è a sola scrittura e viene utilizzato per abilitare o disabilitare tali sorgenti.

E' ovvio, quindi, che se desideriamo utilizzare l'allarme, lo dovremo abilitare con il semplice comando:
POKE 56333,132
mentre nel caso desideriamo disabilitarlo sarà sufficiente:
POKE 56333,4

Abbiamo detto precedentemente che nel caso in cui i due orari (quello in continuo aggiornamento e quello dell'allarme) siano uguali, il computer, oltre che settare il bit 2 del registro 13, attiva anche un interrupt.

Soffermiamoci un momento sulla funzione svolta da quest'ultimo.

L'interrupt è un segnale inviato ogni sessantesimo di secondo al microprocessore (il 6510 nel caso del C-64) che obbliga quest'ultimo ad interrompere l'elaborazione l.m. in corso per eseguire la normale routine d'interrupt che parte dall'indirizzo esadecimale $EA31.

Questa si occupa della gestione della tastiera e in genere di tutte quelle funzioni senza le quali il computer non potrebbe colloquiare con l'esterno.

Per modificare tale routine si agisce (con molta cautela e solo con programmi in L.M.) sulle locazioni 788 e 789 che contengono il suo indirizzo di partenza ($EA31).

E' infatti possibile modificare il contenuto di tali locazioni facendo loro "puntare" una zona di memoria che non sia la solita ($EA31), ma contenente, invece, una routine in

N.	REGISTRI (+ indirizzo CIA)						
BIT	8	9	10	11	13	14	15
	DEC. SEC.	SEC.	MIN.	ORE	IRQ	FREQ. HZ.	OROL. ALLAR.
0	D	U	U	U	*	*	*
1	D	U	U	U	*	*	*
2	D	U	U	U	1-ON 0-OFF	*	*
3	D	U	U	U	*	*	*
4	*	D	D	D	*	*	*
5	*	D	D	*	*	*	*
6	*	D	D	*	*	*	*
7	*	*	*	0-AM 1-PM	*	1-50 0-60	0-OROL. 1-ALLA.

Tabella dei valori contenuti da ciascun bit per i registri del CIA.
"D" indica le decine, "U" le unita', mentre "*" indica un contenuto privo d'importanza.

linguaggio macchina appositamente preparata per il nostro scopo.

Così facendo ogni volta che viene generata una richiesta d'interrupt, il computer salterà alla nuova routine (quella puntata da 788-799) che, ad esempio, dovrà controllare se è stato settato il bit 2 del registro 16 (locazione di memoria 56333 in decimale e $DC0D in esadeciamale). In caso affermativo dovrà produrre, ad esempio, un segnale acustico per avvertire che l'allarme è scattato; quindi salterà alla normale routine d'interrupt posta ad $EA31.

In caso negativo dovrà saltare direttamente ad $EA31 senza eseguire la routine di segnalazione.

Il programma

Come abbiamo già detto, il programma di queste pagine ha solo una funzione dimostrativa.

Subito dopo aver dato il RUN verrà chiesto di introdurre l'orario.

Per fare ciò occorre seguire una certa procedura; prima di tutto dovete rispondere se siamo di mattina (AM) o di pomeriggio (PM). Quindi verrà chiesta la digitazione dell'ora, che ovviamente non deve essere maggiore di 12, e, in seguito, dei minuti e dei secondi.

Una volta terminate le domande verrà costantemente visualizzata l'ora sullo schermo, in alto a sinistra.

Le prime due cifre indicano l'ora, le due successive i minuti, le altre due i secondi e l'ultima i centesimi di secondo (poco utile a causa dell'elevata velocità di scorrimento del tempo rispetto a quella della visualizzazione).

La linea 120 serve per selezionare la frequenza di rete.

Quelle dalla 600 alla 850 si occupano dell'aggiornamento dell'orario e richiamano molto spesso la routine (linee 900-940) che serve per tradurre opportunamente i dati battuti in forma idonea per pokarli nelle apposite locazioni.

Le linee dalla 2000 alla 2215 si occupano di fare l'esatto contrario delle operazioni precedenti, ossia leggono i dati nelle locazioni di memoria del CIA e li trasformano in maniera a noi comprensibile creando una stringa A$ che contiene, per l'appunto, l'ora in forma leggibile.

La linea 2220 si occupa di stampare in alto a sinistra sullo schermo la stringa A$.

```
  30 REM OROLOGIO C/64
  70 REM BY FABIO  CESTARI
  90 REM CUSANO MILANINO (MI)
  95 :
 100 C1=56320
 110 REM * SELEZIONE FREQUENZA 5
     0 HZ.*
 120 POKE C1+14,PEEK(C1+14)  OR
      128
 130 :
 140 :
 600 REM * AGGIORNA ORA *
 610 POKE C1+15,PEEK(C1+15)  AND
       127
 620 PRINTCHR$(147)CHR$(18)"AGGI
     ORNAMENTO DELL'ORA"
 630 PRINT"AM O PM";:A$=""
 640 GET A$:IF A$="" THEN 640
 650 IF A$="A" THEN A=0:PRINT" A
     ":GOTO 670
 660 A=128:PRINT" P"
 670 A$="":INPUT "ORA      ";A$
 680 IF LEN(A$)>2 THEN PRINTCHR$
     (145);:GOTO 670
 690 GOSUB 900
 700 IF Z>18 THEN PRINTCHR$(145)
     ;:GOTO 670
 710 POKE C1+11,Z+A
 720 REM  * AGGIORNA MINUTI *
 730 A$="":INPUT "MINUTI ";A$
 740 IF LEN(A$)>2 THEN PRINTCHR$
     (145);:GOTO 730
 760 GOSUB 900
 770 IF Z>89 THEN PRINTCHR$(145)
     ;:GOTO 730
 780 POKE C1+10,Z
 790 REM * AGGIORNA SECONDI *
 800 A$="":INPUT "SECONDI";A$
 810 IF LEN(A$)>2 THEN PRINTCHR$
     (145);:GOTO 800
 820 GOSUB 900
 830 IF Z>89 THEN PRINTCHR$(145)
     ;:GOTO 800
 840 POKE C1+9,Z:POKE C1+8,0
 850 GOTO 2000
 900 REM * CALCOLO DATI PER LE P
     OKE *
 910 IF LEN(A$)=1 THEN X=0:GOTO
     930
 920 X=VAL(LEFT$(A$,1))
 930 Y=VAL(RIGHT$(A$,1))
 940 Z=16*X+Y:RETURN
2000 REM  OROLOGIO
2010 PRINTCHR$(147):PRINT"  HHMM
     SSC"
2120 A$="":IF (PEEK(C1+11) AND 1
     28)=128 THEN A$="P ":GOTO 2
     135
2130 A$="A "
2135 B$=STR$((PEEK(C1+11) AND 24
     0)/16)
2140 IF VAL(B$)=8 THEN A$=A$+"0"
     :GOTO 2155
2145 IF VAL(B$)=9 THEN A$=A$+"1"
     :GOTO 2155
2150 A$=A$+RIGHT$(B$,1)
2155 B$=STR$(PEEK(C1+11) AND 15)
2160 A$=A$+RIGHT$(B$,1)
2165 B$=STR$((PEEK(C1+10) AND 24
     0)/16)
2170 A$=A$+RIGHT$(B$,1)
2175 B$=STR$(PEEK(C1+10) AND 15)
2180 A$=A$+RIGHT$(B$,1)
2185 B$=STR$((PEEK(C1+9) AND 240
     )/16)
2195 A$=A$+RIGHT$(B$,1)
2200 B$=STR$(PEEK(C1+9) AND 15)
2205 A$=A$+RIGHT$(B$,1)
2210 B$=STR$(PEEK(C1+8) AND 15)
2215 A$=A$+RIGHT$(B$,1)
2220 PRINTCHR$(19)A$
2230 GET C$:IF C$<>"" THEN 140
2235 GOTO 2120
2240 END
```

computer service

VENDITA PER CORRISPONDENZA

ACCESSORI PER COMPUTER COMMODORE

GRUPPO CONTINUITÀ
Fornito senza le 12 batterie a stilo ricaricabili. Consente il funzionamento del Vostro computer Commodore C64 o VIC 20 in assenza di corrente. Durata di funzionamento 30 minuti. Ricarica tramite alimentatore Commodore.

KIT ALLINEAMENTO TESTINA
Composto dal cacciavite, nastro di controllo e strumento di taratura con monitor audio permette il perfetto allineamento dei registratori digitali anche con nastri commerciali.

VELOCIZZATORE DI CARICAMENTO FLOPPY
Cartridge con un insieme di utility residenti su ros per velocizzare il drive nel Commodore 64.

INTERFACCIA RADIO
Indispensabile per registrare con registratore Commodore modello "C2N" i programmi speciali per computer trasmessi dalle emittenti radio.

CUFFIA PER COMMODORE C 64
Leggerissima permette l'ascolto personale del computer evitando di distrurbare durante i giochi.

COPIATORE PROGRAMMI
Dispositivo hardware per effettuare copie di nastri protetti o turbo utilizzando due registratori Commodore o compatibili.

DUPLICATORE CASSETTE
Indispensabile per realizzare delle copie, con un registratore normale, di un nastro protetto o con caricamento turbo.

Articolo	Art.	L.
Bus quadrislot	Art. CD 100	L. 55.000
Interfaccia cassette	Art. CD 101	L. 30.000
Duplicatore cassette	**Art. CD 102**	**L. 30.000**
Copiatore programmi	**Art. CD 103**	**L. 30.000**
Interfaccia radio	**Art. CD 104**	**L. 30.000**
Kit allineamento testina	**Art. CD 105**	**L. 47.000**
Alimentatore per C64 e VIC 20	Art. CD 106	L. 45.000
Gruppo continuità (fornito senza le 12 batterie a stilo ricaricabili)	**Art. CD 107**	**L. 66.000**
Pacco batterie (12 stilo 1,2 Volt ricaricabili	Art. CD 117	L. 52.000
Commutatore antenna TV/computer	Art. CD 108	L. 9.500
Tasto reset	Art. CD 109	L. 5.500
Interfaccia Centronics	Art. CD 112	L. 104.000
Espansione di memoria per C 16	Art. CD 114	L. 158.000
Velocizzatore di caricamento flop,	**Art. CD 115**	**L. 49.000**
Espansione di memoria per VIC 20 16K	Art. CD 116	L. 112.000
Modulatore Executive	Art. CD 120	L. 72.000
Penna ottica grafica	Art. CD 121	L. 45.000
Tavoletta grafica	Art. CD 130	L. 238.000
Multipresa con filtro - 2 prese	Art. CD 140	L. 41.000
Cuffia per Commodore C 64	**Art. CD 150**	**L. 19.000**
Stabilizzatore elettronico di tensione 500 W	Art. CD 160	L. 430.000
Gruppo di cintinuità 60 W	Art. CD 170	L. 400.000
Gruppo di continuità 200 V	Art. CD 180	L. 802.000
Inventer 12 Volt cc. 220 Volt ca. 100 Watt	Art. CD 190	L. 297.000
Cavo alimentazione	Art. CD 200	L. 4.600
Cavo drive o stampante Commodore	Art. CD 205	L. 8.500
Prolunga per Joystick - mt. 3	Art. CD 210	L. 25.000
Prolunga per cavo TV - mt. 3	Art. CD 215	L. 12.500
Cavo audio - mt. 6	Art. CD 220	L. 15.500
Adattatore Joistick (Atari e C64 al C 16)	Art. CD 225	L. 10.500
Adattatore registratore per C 16	Art. CD 226	L. 19.500
Mascherina antiriflesso 12"	Art. CD 300	L. 35.000
Nastro inchiostrato per Tally - mt. 80	Art. CD 610	L. 16.500
Nastro inchiostrato per Tally - mt. 180	Art. CD 611	L. 16.500
Nastro inchiostrato per Tally 1000 e Honeywell	Art. CD 612	L. 9.500
Nastro inchiostrato per Commodore MRS 801	Art. CD 614	L. 13.000
Nastro inchiostrato per Commodore MPS 802	Art. CD 616	L. 18.000
Nastro inchiostrato per Commodore MPS 803	Art. CD 618	L. 19.500
Mause per Commodore C 64	Art. CD 860	L. 240.000
Pacco carta lettura facilitata 24" x 11" modulo da 500 fogli con bordi a strappo	Art. CD 630	L. 13.500
Supporto stampante porta carta in plexiglass "fume" - normale	Art. CD 660	L. 59.000
Supporto stampante porta carta in plexiglass "fume" - rinforzato	Art. CD 670	L. 80.000
Floppy disk 5" singola faccia doppia densità "ODP" - conf. 10 pezzi	Art. CD 700	L. 40.000
Floppy disk 5" singola faccia doppia densità "CBS" - conf. 10 pezzi	Art. CD 702	L. 38.000
Floppy disk 5" singola faccia doppia densità "VERBATIM" - conf. 10 pezzi	Art. CD 704	L. 42.000
Floppy disk 5" singola faccia doppia densità "DYSAN" - conf. 10 pezzi	Art. CD 706	L. 68.000
Nastri magnetici C 10 digitali - conf. 10 pezzi	Art. CD 712	L. 20.000
Nastri magnetici C 15 digitali	Art. CD 714	L. 21.000
Copritastiera in plexiglass per C64 - C16 e VIC 20	Art. CD 750	L. 16.000
Copritastiera in stoffa per C64 - C16 e VIC 20	Art. CD 760	L. 10.500
Vaschetta portafloppy in plexiglass per 40 dischi con chiave	Art. CD 770	L. 30.000
Vaschetta portafloppy in plexiglass per 90 dischi con chiave	Art. CD 780	L. 37.000
Kit pulizia testine registratore	Art. CD 815	L. 13.500
Kit pulizia disk drive	Art. CD 820	L. 26.000
Kit pulizia tastiera	Art. CD 830	L. 16.500
Foratore disk in plastica (per utilizzare la seconda faccia dei dischi)	Art. CD 840	L. 10.000
Foratore disk in metallo "tako"	Art. CD 849	L. 14.000
Joystick Spectravideo II	Art. CD 850	L. 27.000
Joystick a Microswitch	Art. CD 851	L. 52.500
Joystick senza fili con unità ricevente (funziona a batteria)	Art. CD 852	L. 98.000
Joystick per Commodore 16 (originale)	Art. CD 130	L. 29.500

TUTTI I PREZZI SONO COMPRENSIVI DI IVA
NON SI ACCETTANO ORDINI INFERIORI A L. 30.000
CONTRIBUTO FISSO SPESE DI SPEDIZIONE L. 5000

SI ACCETTANO ANCHE ORDINI TELEFONICI AI NUMERI 0522/661647-661471

BUONO DI ORDINAZIONE

NOME - COGNOME

INDIRIZZO — N

C.A.P. — CITTÀ — PROVINCIA

VOGLIATE INVIARMI IN CONTRASSEGNO

N.	Art.	L.
N.	Art.	L.
N.	Art.	L.
SPESE SPEDIZIONE		L. 5.000
PAGHERÒ AL POSTINO		L.

COMPUTER SERVICE VIA A. MANZONI, 49 - 42017 NOVELLARA (RE) - TEL. (0522) 661647

Supertastiera

Come utilizzare il tastierino numerico del C/128 anche in "Modo 64"

di M.Maggi e U.Colapicchioni

Nonostante il trucco sia noto da tempo, grazie ad una delle (poche) notizie diffuse dalla stessa Commodore a proposito del C/128, numerosi sono i lettori che hanno richiesto la procedura da seguire per servirsi della parte destra della tastiera.

Abbiamo quindi deciso di pubblicare una versione sufficientemente breve per aggiungere al C/128, funzionante in modo 64, la comoda possibilità.

Il problema

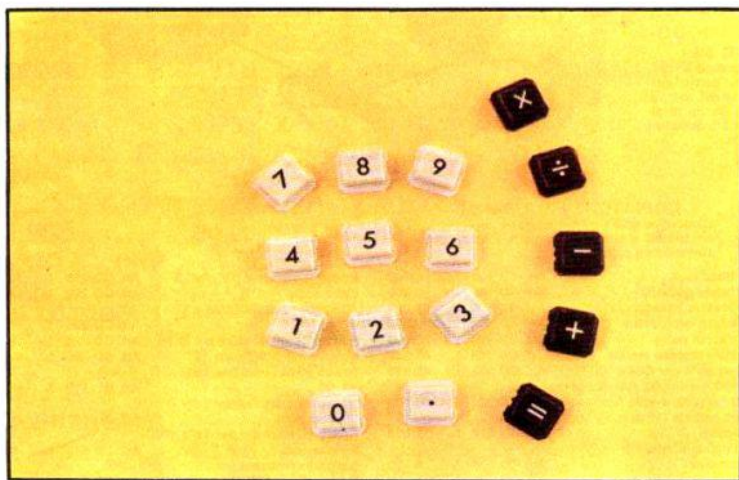
La tastiera del C/128 è formata da ben 92 tasti che sono purtroppo completamente disponibili solo in modo 128.

E' impossibile infatti usare in modo 64 il tastierino numerico che, peraltro, risulta molto comodo se si devono inserire lunghe liste di numeri come quando, ad esempio, si utilizza uno spreadsheet o si deve ricopiare, come in questo caso, una lunga serie di DATA.

Sembrerebbe che non ci sia modo di utilizzare il tastierino in modo 64.

Esiste però una locazione che controlla la scansione di tutti i tasti disponibili.

Questa locazione non è ovviamente presente sul vecchio C/64, ma solo nel 64 "contenuto" all'interno del C/128, ed esattamente all'indirizzo esadecimale $D02F (decimale 53295).

Quando siamo in modo 64, la scansione della tastiera avviene esattamente come nel 64 "normale", ma è possibile modificarla con una routine in linguaggio macchina che estende la scansione a tutta la tastiera, comprendendo anche il tastierino numerico e i quattro tasti cursore.

Tale routine non attiva comunque i tasti Alt, Caps Lock, Tab, Esc, Help, Line feed, 40/80 display, No scroll.

Per utilizzare la routine è necessario ricopiare attentamente tutti i DATA che compongono il listato di queste pagine.Se non avete inserito correttamente i dati il programma darà una segnalazione di errore e sarà necessario controllarlo con attenzione.

Una volta comparsa la scritta che avvisa l'attivazione dei tasti speciali, potremo contare su 18 nuovi tasti da utilizzare sia durante la stesura di programmi, sia durante un qualsiasi inserimento di dati numerici.

Precauzioni da prendere

La routine altera i normali valori dell'interrupt, cioè quella routine che viene richiamata dal Sistema Operativo ogni 60mo di secondo.

Se quindi cercate di utilizzarla con programmi che a loro volta modificano i vettori di interrupt, è probabile che si verifichi un Crash del sistema, costringendo a spegnere, riaccendere e ricominciare daccapo.

Potranno sorgere problemi anche con giochi contenenti musica in sottofondo, e con programmi su cartidge, i quali riportano i vettori dell'interrupt a valori standard.

Un'altra limitazione è dovuta al fatto che premendo i tasti Run/Stop e Restore la routine stessa viene disattivata: per ripristinarla è comunque sufficiente un SYS 976.

Non dimenticate, inoltre, che la routine occupa la zona di memoria riservata alla gestione del nastro cassetta. Se dovete utilizzare il registratore, quindi, escludetela mediante Run/Stop e Restore e in seguito, una volta completate le operazioni su nastro, ricaricatela e lanciatela.

Chi utilizza il drive, invece, non avrà questo problema.

```
100 REM  SUPERTASTIERA
110 REM  PER C/128
120 REM  IN MODO 64
130 :
140 REM  BY M.MAGGI & U. COLAPI
    CCHIONI
150 :
160 TEMS=976
170 B=828:PRINTCHR$(147)"STO LE
    GGENDO I DATI...."
180 READ A:POKE B,A:B=B+1:C=C+A
    :IF B=974 THEN B=B+1:GOTO 1
    80
190 IF B<>989 THEN 180
200 IF C=18512 THEN SYSTEMS:PRI
    NTCHR$(147)"TASTIERINO NUME
    RICO IN FUNZIONE":END
210 PRINT"ERRORE NEI DATA":END
220 DATA 169,3,72,169,75,72,8,7
    2,165,197,72
230 DATA 72,76,49,234,120,160,0
    ,165,203,201,64
240 DATA 208,88,169,255,141,0,2
    20,140,47,208
250 DATA 173,1,220,201,255,240,
    73,134,197,169
260 DATA 254,72,162,8,141,47,20
    8,173,1,220,205,1,220,208,2
    48,74,176,9,72,185,183
270 DATA 3,240,2,133,203,104,20
    0,202,208,240,104,56,42,192
    ,23,144,219,165,203
280 DATA 201,64,240,26,162,129,
    160,0,144,8,41,127,133,203,
    162,194,160,1,169,235
290 DATA 140,141,2,134,245,133,
    246,32,224,234,169,255,141,
    47,208,32,66,235,76
300 DATA 129,234,0,27,16,0,59,1
    1,24,56,0,40
310 DATA 43,0,1,19,32,8,0,35,44
    ,135,7,130,2,0,120
320 DATA 169,60,141,20,3,169,3,
    141,21,3,88,96
```

MANNESMANN TALLY

20094 Corsico (MI) - Via Borsini, 6
Tel. (02) 4502850/855/860/865/870
Telex 311371 Tally I
00144 Roma - Via M. Peroglio, 15
Tel. (06) 5984723/5984406
10099 San Mauro (TO)
Via Casale, 308 - Tel. (011) 8225171
40050 Monteveglio (BO)
Via Einstein, 5 - Tel. (051) 832508

adverteam

Finalmente su disco l'Enciclopedia delle Routine

Software a basso costo, programmi sprotetti e listabili, possibilità di modificarli; ed altro ancora...

di Alessandro de Simone

Come già accennato nell'editoriale, pubblichiamo la Directory (da cui il nome della nuova iniziativa) del dischetto che è possibile richiedere in Redazione secondo le modalità indicate in fondo all'articolo.

I listati pubblicati sono piuttosto numerosi e tra questi figurano TUTTE le routine della nota "Enciclopedia" che tanto successo ha riscosso presso gli utenti Commodore.

Si noti, inoltre, la presenza di TUTTI i listati pubblicati sul numero scorso e, se non bastasse, gli altri programmi che costituiscono una compilation di Best Seller di giochi comparsi sulla rivista su cassetta "Software Club".

Tutti i programmi sono listabili e copiabili, tranne che per scopi commerciali. Gli appassionati possono quindi scambiarli tra loro, MA NON VENDERLI. La Systems Editoriale tutelerà i propri interessi nei confronti degli umanoidi che, nonostante tutto, decidessero di inserire queste routine nelle proprie cassette pirata.

I programmi in linguaggio macchina, ovviamente, sembrano costituiti da un'unica SYS di partenza. Gli esperti, caricando un Monitor, potranno, comunque, disassemblarli per esaminarli con comodo a volontà.

L'intervento dei lettori

La novità principale, però, consiste nel sollecitare il lettore a sofisticare alcuni dei programmi inseriti in "Directory" e ad inviarne una nuova versione. Questa, dotata, ad esempio, di più opzioni, oppure più veloce o, in una parola, "migliore" di quella contenuta nel dischetto che proponiamo, verrà inserita in un numero successivo di "Directory" disponibile per ulteriori miglioramenti da parte di altri lettori; e così via in una gara che se, da una parte, consente la diffusione di software via via più potente, dall'altra costituisce una validissima palestra per quei lettori che non chiedono di meglio che migliorare la propria bravura alla tastiera.

Come primo "esercizio" suggeriamo di razionalizzare il programma "Drums (batteria elettronica)" che è scritto in Basic ma risulta, in verità, piuttosto... disordinato: Applicate, ad esempio, i suggerimenti dell'inserto "La programmazione modulare" del N.34 di C.C.C.

Visto che ci siete, poi, provate ad inserire qualche altra opzione come, ad esempio, la possibilità di memorizzare due ritmi e di eseguire il primo, o il secondo, a seconda della pressione di un tasto.

Come procurarsi "Directory"

Avvertiamo i lettori che NON è assolutamente possibile inviare i programmi su nastro, per intuibili motivi di economia ed affidabilità del nastro cassetta.

"Directory" può quindi esser richiesta solo su disco inviando L.12000 + 3000 per le spese di spedizione.

Non ci è possibile inviare materiale contrassegno.

Compilate un normale modulo di C/C postale indirizzando a:

C/C postale N. 37952207
Systems Editoriale
Viale Famagosta, 75
20142 Milano

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento") non solo il vostro nominativo completo, ma anche il nome del disco desiderato:

"Directory N.1"

N.B. Per ottenere il materiale in tempi più ristretti, inviate l'importo a mezzo assegno bancario non trasferibile: le poste italiane non brillano per velocità!

Altri suggerimenti

I lettori, però, possono collaborare a "Directory" inviando listati, file e altri... "oggetti" informatici che non è possibile pubblicare su riviste di carta. Ecco alcuni esempi, tra i primi che ci vengono in mente:

- Set di caratteri ridefiniti che avete realizzato per vostri scopi particolari e che nessuna rivista del settore avrà mai il coraggio di pubblicare (immaginate la digitazione di 4K (e passa) di DATA?)
- Realizzazione di più sprite da far apparire in successione per animazioni.
- Disegni particolarmente pittoreschi eseguiti col Koala, o con altri Tool grafici, da caricare in propri programmi.
- Archivi di ogni tipo, realizzati ricorrendo a programmi commercializzati o a listati di vostra creazione. Esempi: archivi di tutte le colonne vincenti del Totocalcio dal 1980 ad oggi; archivi degli articoli apparsi su Commodore Computer Club suddivisi in più campi (Giochi, Hardware, Protezioni, Grafica, Musica eccetera) da

utilizzare in unione con "Superbase", oppure "The Manager" o altri ancora.

• Vocabolari di centinaia di parole, facilissimi da realizzare e da utilizzare con un Word Processor: ricorrendo alla funzione Search (oppure Find) presente in qualsiasi W/P, è infatti possibile rintracciare la parola cercata (a patto che qualcuno abbia realizzato il file-vocabolario...)

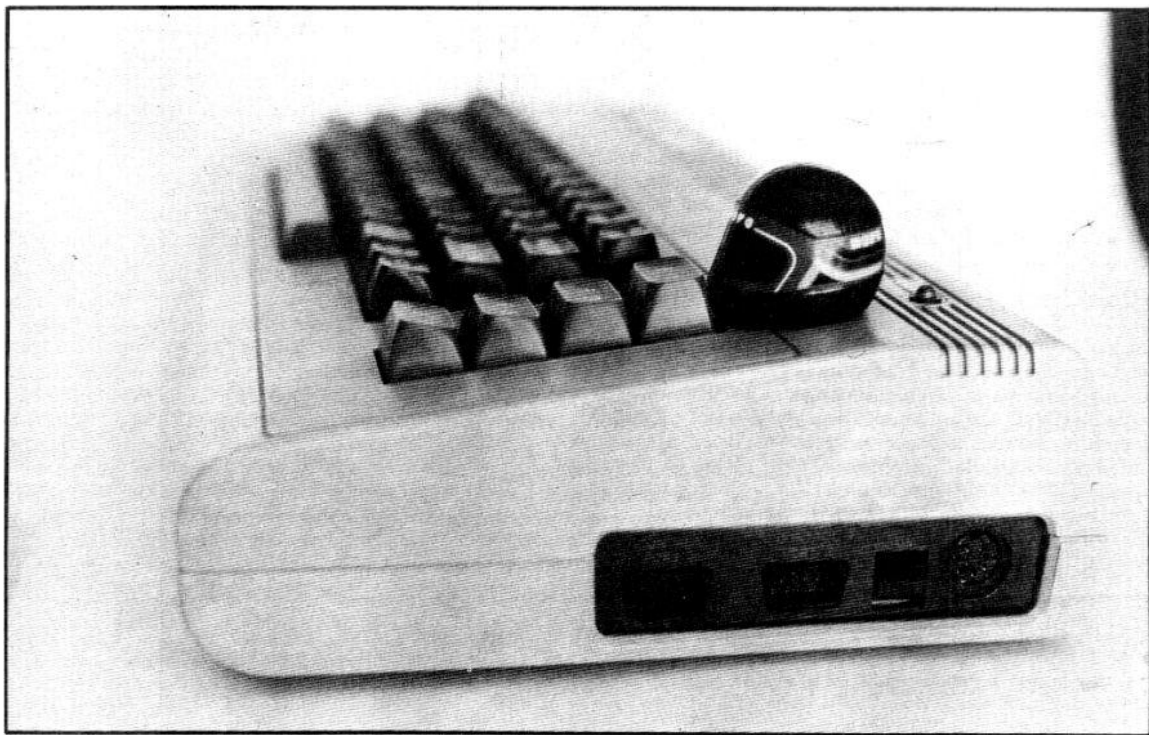
Centinaia sono comunque le occasioni per partecipare a "Directory".

Prima di inviare il risultato del vostro lavoro vi consigliamo, però, di telefonarci per stabilire se risponde ai requisiti per l'eventuale pubblicazione (tel. 02/8467348)

```
0 "SETTE               " 31 2A
1     "STARTER PROGRAM"   PRG
9     "SPIEGAZIONI"       SEQ
102   "CAGIVA GAME"       PRG
56    "GHEDDAFIAH"        PRG
48    "CROCKET"           PRG
50    "NEMO"              PRG
3     "100 CORNIC POL24"  PRG
3     "101 LAMP.VIDEO24"  PRG
4     "102 INDIVID PA24"  PRG
4     "103 INPUT DEF.25"  PRG
3     "500 BLOCKS FRE25"  PRG
4     "104 INCOL.VIRG25"  PRG
4     "105 INPUT CONT25"  PRG
4     "106 REVER SCRE25"  PRG
3     "107 IMPUL/SONO25"  PRG
4     "108 CONTR.DATA25"  PRG
4     "109 DECIM/ESA 26"  PRG
3     "110 FUNZ IPERB26"  PRG
3     "111 FUNZ INVER26"  PRG
3     "112 INVER/TRIG26"  PRG
4     "113 INV/IPERB.26"  PRG
5     "501 ESAM.DIREC26"  PRG
4     "114 ISTOGRAMM 27"  PRG
4     "115 CAR HI-RES27"  PRG
4     "116 SC/SILLABE27"  PRG
4     "117 MICROCALC.27"  PRG
6     "118 SAVE MEMOR27"  PRG
5     "119 SCAM.VIDEO27"  PRG
5     "120 KOALA VIDE27"  PRG
3     "121 PROTEZIONE28"  PRG
4     "122 NUM.CONGR.28"  PRG
3     "500 BL/FRE/2/28"   PRG
6     "502 CREA RELAT28"  PRG
5     "503 SCRIVE REL28"  PRG
4     "504 LEGGE REL 28"  PRG
4     "505 DISP.FILE 28"  PRG
4     "123 MCD MCM   29"  PRG
5     "124 PRINT USIN29"  PRG
4     "126 TEXT COPY 29"  PRG
4     "125 CAMBIA COL29"  PRG
5     "127 FILL MEMO 29"  PRG
5     "128 BORDO COLO29"  PRG
3     "129 SCRIT.LAMP29"  PRG
4     "130 SCELTA MEN30"  PRG
5     "131 SCELTA JOY30"  PRG
5     "124 P/USING/2 31"  PRG
4     "133 ELAB/STRIN31"  PRG
6     "134 PLOT LOW/R31"  PRG
5     "135 COMANDI FP31"  PRG
4     "136 FUNZ. EOR 31"  PRG
6     "137 BIT IMAGE 31"  PRG
5     "138 EXTE/BACK.31"  PRG
4     "139 SCREEN PAG31"  PRG
4     "132 CENTR/MESS32"  PRG
4     "140 NOME DISK 32"  PRG
6     "141 FINESTRE  32"  PRG
4     "142 OROLOGIO  33"  PRG
3     "143 ZOOM ESAD 33"  PRG
5     "144 SPRITE M/U33"  PRG
4     "145 SCROLL TEX33"  PRG
4     "105 INPUT PROG34"  PRG
3     "146 RUOTA STR.34"  PRG
3     "147 SLITTA ST.34"  PRG
3     "148 SOST.STRI.34"  PRG
4     "149 CANC/WIND.35"  PRG
4     "150 FRAM/SCHER35"  PRG
4     "151 MESS/LAMP.35"  PRG
42    "DRUMS (BATTERIA)"  PRG
3     "CARICAMI ......."  PRG
9     "CARATTERI"         PRG
2     "TURBO SORT PRG"    PRG
3     "AUTORUN PROGRAMM"  PRG
10    "CRIPTO PROG"       PRG
3     "SCROLL CARATTERE"  PRG
2     "DEMO SCROLL FLG"   PRG
2     "DEEK L/M"          PRG
3     "DEMO DEEK"         PRG
2     "DOKE L/M"          PRG
1     "DEMO DOKE"         PRG
25    "ATTERRAGG/LUN"     PRG
3     "PROGRAMMA MOUSE"   PRG
2     "PRG PADDLE VIC 2"  PRG
34 BLOCKS FREE.
```

Scrambler (Cripto/2)

Una versione diversa e più breve di quella pubblicata in precedenza

di Roberto Morassi

Non appena pubblicato sul numero scorso il programma di protezione "Cripto", un nostro lettore ha subito provveduto ad inviarne una versione che, pur basata sullo stesso sistema, è molto più breve e si fonda sull'utilizzo della funzione RND(). Riepiloghiamo, comunque, un po' di teoria per i lettori che avessero dimenticato in che cosa consiste il metodo.

Un po' di teoria

Uno dei possibili metodi di protezione di un programma Basic è il cosiddetto "Scrambling", vale a dire la modifica intenzionale (e reversibile) di tutti i byte del programma.

Più che di una protezione, si tratta di una "cifratura" che rende il programma non listabile nè eseguibile se non dopo adeguata "decifrazione" che ripristini tutti i byte originali: questa implica, ovviamente, la conoscenza del modo con cui il programma è stato cifrato e di eventuali "chiavi" di accesso.

Il metodo standard di "scrambling" è l'OR-esclusivo (EOR) dei singoli byte del programma con un byte di riferimento ("maschera").

L'operatore EOR confronta i bit corrispondenti di due byte, fornendo "in uscita" un terzo byte i cui bit valgono "1" se i due bit confrontati sono uno "0" e un "1", oppure "0" se i due bit confrontati sono entrambi "0" oppure entrambi "1".

Tale operatore, se applicato una seconda volta, riporta i byte nel loro

stato iniziale; è facile verificare che se A EOR B=C, allora C EOR B=A: ciò significa che la routine usata per cifrare il programma può essere usata tal quale per decifrarlo.

A tale scopo è ovviamente necessario conoscere il valore che è stato usato come "maschera" per l'OR-esclusivo. In teoria, però, tale valore si potrebbe ricavare per tentativi, provando sistematicamente tutti i valori da 1 a 255 fino ad ottenere un LIST corretto.

La versione proposta

La routine SCRAM di queste pagine effettua lo scrambling con una variante del metodo descritto, leggermente più... perfida: l'EOR viene fatto non con un solo byte-maschera, ma con quattro byte diversi che si alternano ciclicamente.

Questi vengono letti nei registri 140-143, che contengono, di norma, il "seme" usato dall'istruzione Basic RND: poichè tale seme è determinato univocamente se si è eseguita un'istruzione RND con argomento negativo, si ha una chiave di accesso personalizzata senza la quale non sarà possibile ricostruire il programma cifrato, neppure avendo SCRAM a disposizione. Vediamo come si usa in pratica.

Come usare Scram

- Digitate e salvate SCRAM, poi date il RUN per caricarlo in memoria a partire dal registro 49152 (la routine è comunque completamente rilocabile senza modifiche).

- Date il NEW, poi caricate in memoria il programma che volete cifrare.

- Date il comando X=RND(-N), sostituendo ad "N" un numero, intero o decimale, a vostra scelta: questo rappresenterà la vostra "chiave" personale di decifrazione (attenti a non dimenticarla!).

- Infine digitate SYS 49152. Potete verificare con un LIST che il vostro programma è stato trasformato in una sequenza di byte privi di senso. Ripetete SYS 49152, ed ecco lì il vostro programma pronto a funzionare!

- Con SYS 49152 cifrate di nuovo il programma, e salvatelo in questa forma su supporto magnetico.

La decodifica

Per usare nuovamente il programma "protetto", è necessario ripetere tutte le operazioni precedenti e cioè:

- ricaricare il programma dopo aver messo SCRAM in memoria.
- eseguire X=RND(-N) con lo stesso N di prima.
- impartire il comando SYS 49152.

Come funziona Scram

Per quanto riguarda il funzionamento di SCRAM, è da notare che solo gli ultimi byte eseguono lo scrambling vero e proprio (cioè l'EOR ciclico). Tutto il resto del programma è dedicato alle seguenti operazioni:

- Esclusione dallo scrambling degli "zero" di fine linea Basic e dei due byte successivi, che costituiscono il "link" tra le linee Basic;

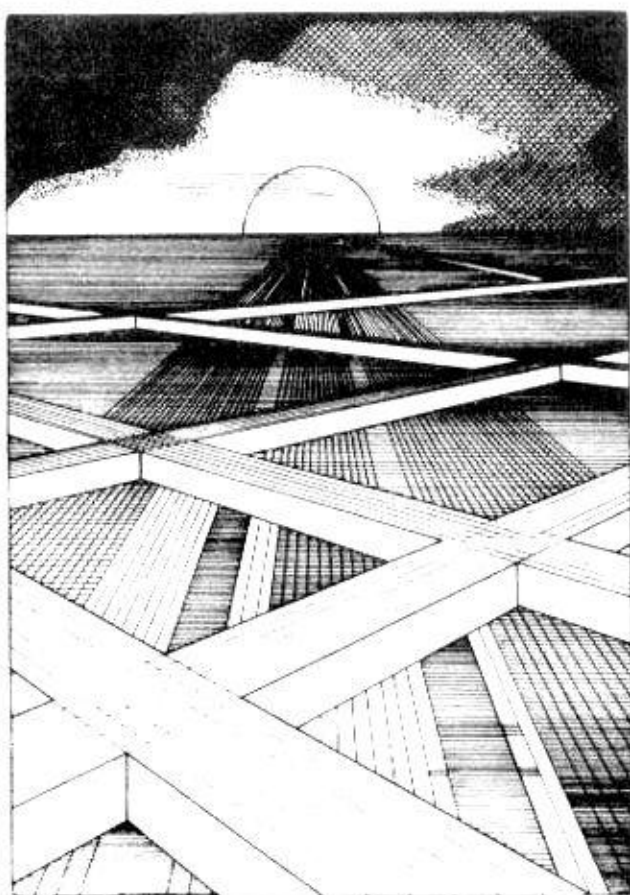
- Inibizione degli "zero" che, eventualmemte presenti nei byte che contengono i numeri di linea, vengano scambiati per segnali di fine linea.

- Impedimento, infine, che eventuali "zero" generati dall'EOR vengano introdotti nel programma cifrato.

Queste precauzioni sono necessarie per garantire che il "Relink" delle linee Basic, che segue automaticamente il caricamento del programma da disco o nastro, funzioni nel modo corretto: in caso contrario verrebbero introdotti nel programma dei byte casuali e imprevedibili che ne renderebbero definitivamente impossibile la decifrazione.

```
 10 REM SCRAMBLER
 20 :
 30 REM PROTEZIONE PER C/64
 35 :
 40 REM BY ROBERTO MORASSI
 50 REM PISTOIA
 60 :
120 :
130 FOR X=0 TO 81:READ Y:POKE 4
    9152+X,Y:A=A+Y:NEXT
140 IF A<>12921 THEN PRINT"ERRO
    RE NEI DATA"
150 END
160 DATA  165,043,133,251,165,0
    44,133,252
170 DATA  162,004,160,255,132,2
    50,200,230
180 DATA  251,208,002,230,252,1
    60,000,230
190 DATA  250,230,251,208,002,2
    30,252,056
200 DATA  165,251,229,045,165,2
    52,229,046
210 DATA  144,001,096,177,251,2
    08,014,164
220 DATA  250,192,002,048,008,2
    30,251,208
230 DATA  209,230,252,208,205,0
    85,139,208
240 DATA  006,164,250,192,002,0
    16,004,160
250 DATA  000,145,251,202,208,1
    99,162,004
260 DATA  208,195
```

La programmazione modulare: *i listati*

(vedi CCC n. 34)

di Alessandro de Simone

PRIMO ESEMPIO

```
 100 REM  PRIMO ESEMPIO
 150 :
 200 X$="":REM  STRINGA DI COMODO
 250 X=0:REM  ADDENDO O MINUENDO
 300 Y=0:REM  ADDENDO O SOTTRAENDO
 350 :
 400 PRINTCHR$(147)
 450 PRINT"1/ ADDIZIONE":PRINT"2/ SOTTRA
     ZIONE"
 500 PRINT"CHE COSA VUOI FARE?"
 550 GET X$:IF X$="" THEN 550
 600 IF X$="1" THEN 800
 650 IF X$="2" THEN 1000
 700 GOTO 400
 750 :
 800 INPUT "PRIMO ADDENDO";X
 850 INPUT "SECON.ADDENDO";Y
 900 PRINT"LA SOMMA VALE"X+Y:END
 950 :
1000 INPUT "MINUENDO";X
1050 INPUT "SOTTRAENDO";Y
1100 PRINT"DIFFERNZA="X-Y:END
```

SECONDO ESEMPIO

```
1000 REM  SECONDO ESEMPIO
1050 :
1100 X$="":REM  STRINGA DI COMODO
1150 X=0:REM  ADDENDO O MINUENDO
1200 Y=0:REM  ADDENDO O SOTTRAENDO
1250 AS=0:REM  CODICE ASCII TASTO
1300 :
1350 PRINTCHR$(147)
1400 PRINT"1/ ADDIZIONE":PRINT"2/ SOTTRA
     ZIONE"
1450 PRINT"3/ FINE LAVORO"
1500 :
1550 GOSUB 2600
1600 IF X$="1" THEN GOSUB 1850:GOTO 1350
1650 IF X$="2" THEN GOSUB 2200:GOTO 1350
1700 IF X$="3" THEN END
1750 GOTO 1350
1800 :
1850 INPUT "PRIMO ADDENDO";X
1900 INPUT "SECON.ADDENDO";Y
1950 PRINT"LA SOMMA VALE"X+Y
2000 GOSUB 2550
2050 IF AS=13 THEN RETURN
2100 GOTO 1850
2150 :
2200 INPUT "MINUENDO";X
2250 INPUT "SOTTRAENDO";Y
2300 PRINT"DIFFERENZA="X-Y
2350 GOSUB 2550
2400 IF AS=13 THEN RETURN
2450 GOTO 2200
2550 PRINT"TASTO RETURN=MENU"
2600 PRINT"PREMI IL TASTO"
2650 GET X$:IF X$="" THEN 2650
2700 AS=ASC(X$):RETURN
```

TERZO ESEMPIO

```
1000 REM  TERZO ESEMPIO
1050 :
1100 X$="":REM  STRINGA DI COMODO
1150 X=0:Y=0:REM  ADDEN/MIN/FATT/DIVID.
1200 AS=0:REM  CODICE ASCII TASTO
1250 SO=0:REM  ULTIMA ELABOR.COMPIUTA
1300 :
1350 REM *** MENU PRINCIPALE ***
1400 PRINTCHR$(147)
1450 PRINT"1/ ADDIZIONE":PRINT"2/ SOTTRA
     ZIONE"
1500 PRINT"3/ MOLTIPLICAZIONE":PRINT"4/
     DIVISIONE"
1550 PRINT"F/ FINE LAVORO":PRINT
1600 GOSUB 3000
1650 IF X$="1" THEN GOSUB 2050:GOTO 1400
1700 IF X$="2" THEN GOSUB 2500:GOTO 1400
1750 IF X$="3" THEN GOSUB 3250:GOTO 1400
1800 IF X$="4" THEN GOSUB 3700:GOTO 1400
1850 IF X$="F" THEN END
1900 GOTO 1400
1950 :
2000 REM *** SOMMA ***
2050 PRINT:INPUT "PRIMO ADDENDO";X$:IF X
     $="SO" THEN X=SO:GOTO 2150
2100 X=VAL(X$)
2150 INPUT "SECON.ADDENDO";Y
2200 SO=X+Y:PRINT"LA SOMMA VALE"SO
2250 GOSUB 2950
2300 IF AS=13 THEN RETURN
2350 GOTO 2050
2400 :
2450 REM *** SOTTRAZIONE ***
2500 PRINT:INPUT "MINUENDO";X$:IF X$="SO
```

```
      " THEN X=SO:GOTO 2600
2550 X=VAL(X$)
2600 INPUT "SOTTRAENDO";Y
2650 SO=X-Y:PRINT"DIFFERENZA="SO
2700 GOSUB 2950
2750 IF AS=13 THEN RETURN
2800 GOTO 2500
2850 :
2900 REM *** ESAME TASTO PREMUTO ***
2950 PRINT"TASTO RETURN=MENU"
3000 PRINT"PREMI IL TASTO"
3050 GET X$:IF X$="" THEN 3050
3100 AS=ASC(X$):RETURN
3150 :
3200 REM *** MOLTIPLICAZIONE ***
3250 PRINT:INPUT "PRIMO FATTORE";X$:IF X
     $="SO" THEN X=SO:GOTO 3350
3300 X=VAL(X$)
3350 INPUT "SECON.FATTORE";Y
3400 SO=X*Y:PRINT"PRODOTTO:"SO
3450 GOSUB 2950
3500 IF AS=13 THEN RETURN
3550 GOTO 3250
3600 :
3650 REM *** DIVISIONE ***
3700 PRINT:INPUT "DIVIDENDO";X$:IF X$="S
     O" THEN X=SO:GOTO 3800
3750 X=VAL(X$)
3800 INPUT "DIVISORE";Y:IF Y=0 THEN 3800
3850 SO=X/Y:PRINT"QUOZIENTE:"SO
3900 GOSUB 2950
3950 IF AS=13 THEN RETURN
4000 GOTO 3700
```

QUARTO ESEMPIO

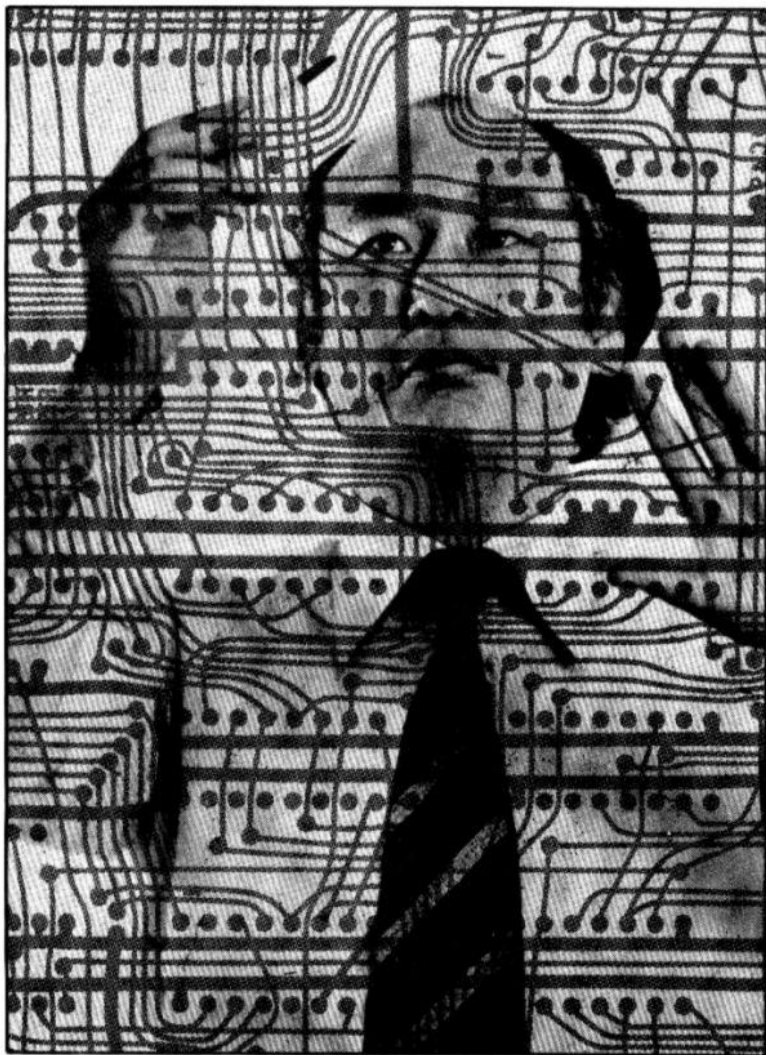
```
1000 REM  QUARTO ESEMPIO
1050 :
1100 X$="":REM  STRINGA DI COMODO
1150 X=0:Y=0:REM  ADDEN/MIN/FATT/DIVID.
1200 Z=0:REM  VARIABILE DI COMODO
1250 AS=0:REM  CODICE ASCII TASTO
1300 SO=0:REM  ULTIMA ELABOR.COMPIUTA
1350 DIM  X(9):REM  VETTORE DI MEMORIE
1400 :
1450 :
1500 REM *** MENU PRINCIPALE ***
1550 PRINTCHR$(147)
1600 PRINT"1/ ADDIZIONE":PRINT"2/ SOTTRA
     ZIONE"
1650 PRINT"3/ MOLTIPLICAZIONE":PRINT"4/
     DIVISIONE"
1700 PRINT:PRINT"A/ AIUTO":PRINT"F/ FINE
      LAVORO":PRINT
1750 GOSUB 3350
1800 IF X$="1" THEN GOSUB 2300:GOTO 1550
1850 IF X$="2" THEN GOSUB 2800:GOTO 1550
1900 IF X$="3" THEN GOSUB 3650:GOTO 1550
1950 IF X$="4" THEN GOSUB 4150:GOTO 1550
2000 IF X$="A" THEN GOSUB 5650:GOTO 1550
2050 IF X$="F" THEN END
2100 GOTO 1550
2150 :
2200 :
2250 REM *** SOMMA ***
2300 PRINT:PRINT"PRIMO ADDENDO";:GOSUB 5
     200
2350 INPUT "SECON.ADDENDO";Y
2400 SO=X+Y:PRINT"LA SOMMA VALE"SO
2450 GOSUB 3300
```

```
2500 IF AS=13 THEN RETURN
2550 IF X$="M" THEN GOSUB 4650
2600 GOTO 2300
2650 :
2700 :
2750 REM *** SOTTIRAZIONE ***
2800 PRINT:PRINT"MINUENDO";:GOSUB 5200
2850 INPUT "SOTTRAENDO";Y
2900 SO=X-Y:PRINT"DIFFERENZA="SO
2950 GOSUB 3300
3000 IF AS=13 THEN RETURN
3050 IF X$="M" THEN GOSUB 4650
3100 GOTO 2800
3150 :
3200 :
3250 REM *** ESAME TASTO PREMUTO ***
3300 PRINT"TASTO RETURN=MENU"
3350 PRINT"PREMI IL TASTO"
3400 GET X$:IF X$="" THEN 3400
3450 AS=ASC(X$):RETURN
3500 :
3550 :
3600 REM *** MOLTIPLICAZIONE ***
3650 PRINT:PRINT"PRIMO FATTORE";:GOSUB 5
     200
3700 INPUT "SECON.FATTORE";Y
3750 SO=X*Y:PRINT"PRODOTTO:"SO
3800 GOSUB 3300
3850 IF AS=13 THEN RETURN
3900 IF X$="M" THEN GOSUB 4650
3950 GOTO 3650
4000 :
4050 :
4100 REM *** DIVISIONE ***
4150 PRINT:PRINT"DIVIDENDO";:GOSUB 5200
4200 INPUT "DIVISORE";Y:IF Y=0 THEN 4200
4250 SO=X/Y:PRINT"QUOZIENTE:"SO
```

```
4300 GOSUB 3300
4350 IF AS=13 THEN RETURN
4400 IF X$="M" THEN GOSUB 4650
4450 GOTO 4150
4500 :
4550 :
4600 REM *** MEMORIZZAZIONE ***
4650 PRINT:FOR Z=0 TO 9:PRINT"MEM."Z;X(Z
     ):NEXT:PRINT
4700 PRINT:INPUT "QUALE MEMORIA (0/9)";X
     :IF X<0  OR  X>9 THEN  RETURN
4750 X(X)=SO:RETURN
4800 :
4850 :
4900 REM *** RICHIAMO MEMORIA ***
4950 PRINT:INPUT "QUALE MEMORIA (0/9)";X
     :IF X<0  OR  X>9 THEN  RETURN
5000 Y=X(X):RETURN
5050 :
5100 :
5150 REM *** ESAME INPUT ***
5200 INPUT X$:IF X$="SO" THEN X=SO:RETUR
     N
5250 IF X$="M" THEN 5350
5300 X=VAL(X$):RETURN
5350 PRINT:FOR Z=0 TO 9:PRINT"MEM."Z;X(Z
     ):NEXT:PRINT
5400 PRINT"QUALE?";:GOSUB 3400:IF X$<"0"
      OR X$>"9" THEN RETURN
5450 PRINTX$:X=X(VAL(X$)):RETURN
5500 :
5550 :
5600 REM *** LETTURA FILE DI AIUTO ***
5650 OPEN 8,8,8,"AIUTO,S,R"
5700 GET X$:IF X$ THEN CLOSE 8:RETURN
5750 IF ST THEN CLOSE 8:RETURN
5800 GET #8,X$:PRINTX$;:GOTO 5700
```

Tutto sugli sprite

E' giunto il momento di conoscere bene gli sprite. Dunque: c'era una volta...

di Pasquale D'Andreti

Lo sprite (folletto, shape, figura o come dir si voglia) è una delle numerose opzioni che il Commodore 64, attraverso il suo chip video VIC II, può gestire.

Ne possono essere presenti fino ad otto contemporaneamente sullo schermo, manovrati indipendentemente l'uno dall'altro e visualizzati con modi grafici diversi.

L'importanza degli sprite nella programmazione di figure in movimento è stata, insieme alle altre possibilità grafiche e sonore del C/64, l'elemento determinante che ha portato questo computer ad avere la più grande disponibilità di software ludico.

Far muovere, infatti, una figura sullo schermo, fornendo solo le coordinate d'arrivo (e senza preoccuparsi per ciò che possa succedere "sotto" lo sprite), è estremamente più facile e rapido che attuarlo via software (si pensi allo Spectrum o al C/16).

La figura animata (sprite) è composta da 24x21 pixel (PIcture ELement= unità figurativa) visualizzabili anche in multicolor ed espansi secondo i due assi cartesiani.

Ma non è finita: il singolo sprite ha, infatti, due tipi di priorità di visualizzazione: quella verso il fondo e quella verso gli altri sprite.

Intervenendo sul primo tipo di priorità si può stabilire se lo sprite deve "coprire" i caratteri su cui passa, oppure se deve essere "nascosto" da questi ultimi.

Il secondo tipo di priorità è, invece, fissa e non si può modificare: riguarda gli otto sprite insieme, regolandone la visualizzazione secondo il numero del singolo sprite: quello di numero più basso coprirà quello di numero più alto.

Lo sprite, inoltre, può essere visualizzato su ogni tipo di schermo, in alta o bassa risoluzione, in multicolor o sui caratteri ridefiniti. Questa ulteriore caratteristica è stata riportata per ultima ma, a nostro avviso, è la più importante.

Procederemo con ordine fornendo, via via, gli elementi necessari alla definizione e all'uso degli sprite.

La costruzione degli sprite

Prima di visualizzare uno sprite lo si deve ovviamente definire, ossia si deve memorizzare, da qualche parte, la sua "immagine" tradotta in un gruppo di numeri.

Come abbiamo già affermato, lo sprite è formato da una matrice di 24 pixel orizzontali e 21 verticali.

La memoria del C/64 è costituita da diverse decine di migliaia di byte i quali, a loro

volta, sono composti da 8 bit ciascuno.

Questi bit possono valere soltanto 0 oppure 1 e possono facilmente essere identificati con il singolo pixel: se il pixel è acceso, il bit corrispondente vale 1, altrimenti vale 0.

A questo punto, facendo riferimento alla figura 1, possiamo notare che la griglia è ulteriormente suddivisa in tre colonne da otto pixel ciascuna.

Ora non ci metteremo certo a trattare la teoria dei numeri in base 2 o binari (del resto l'argomento è stato trattato ampiamente sul numero di febbraio 86 nel primo articolo della serie LA GRAFICA DEL C/64) ma daremo solo indicazioni valide in questo specifico caso.

Il nostro problema è quello di "tradurre" in notazione decimale i gruppi di 8 pixel-bit-cifre che compongono i 63 byte da memorizzare.

```
      byte 1    byte 2    byte 3
     76543210 76543210 76543210
   -+---------+---------+---------+
   1|         | ****    |         |
   2|      ** |******** |         |
   3|    **** |******** |         |
   4|   ***** |******** |         |
   5|  ****** |*     ** |         |
   6| ******  |       * |*******  |
   7| *****   |       * |******   |
   8| ****    |       * |*****    |
   9|*****    |       * |****     |
  10|*****    |       * |***      |
  11|*****    |         |         |
  12|*****    |       * |***      |
  13|*****    |       * |   *     |
  14| ****    |       * |    *    |
  15| *****   |       * |     *   |
  16| ******  |       * |*******  |
  17|  ****** |*     ** |         |
  18|   ***** |******** |         |
  19|    **** |******** |         |
  20|      ** |******** |         |
  21|         | ****    |         |
    +---------+---------+---------+
      byte 61   byte 62   byte 63
```

Figura 1:
Sprite disegnato nella griglia 24x21.

Queste colonne individuano i tre byte occorrenti per ciascuna riga dello sprite; siccome questo è composto da 21 righe, occorreranno 63 (21x3) byte per memorizzarlo.

Riepilogando: per costruire l'immagine di uno sprite, è necessario:

- *disegnare una matrice di 24x21 elementi.*
- *definire i contorni della figura scelta.*
- *annerire le celle corrispondenti.*

In seguito dovremo calcolare i valori dei 63 byte; questi sono espressi in binario in quanto le 8 cifre (bit) costituenti ogni byte possono valere solo 0 e 1.

La regola pratica è molto facile da attuare poichè si basa su un algoritmo semplicissimo.

Basta, infatti, dare una posizione agli otto bit numerandoli da sinistra a destra, cominciando dal numero 7 e decrescendo fino a 0. Poi si considera, per ogni bit, una costante K il cui valore è dato dall'espressione:

K=2↑P

dove P è la posizione del bit trovato prima.

Così facendo troveremo 8 valori che saranno associati ai bit nel modo seguente:

Pos.	Val.
7	128
6	64
5	32
4	16
3	8
2	4
1	2
0	1

L'ultima operazione da effettuare consiste nell'eseguire una somma i cui addendi siano costituiti da tutte e sole le costanti calcolate predentemente la cui posizione contenga un bit di valore uno, ossia una casella del disegno annerita.

Per maggiore chiarezza riferiamoci alla tredicesima riga dello sprite di figura 1 composta, come tutte le altre, da tre byte affiancati.

I valori dei tre byte sono ricavati dalle somme seguenti:

0+64+32+16+8+0+0+0=120
0+0+0+0+0+0+0+1=1
0+0+0+0+8+0+0+0=8

Gli zeri, benchè superflui, sono stati inseriti per meglio evidenziare la funzione che assumono i pixel anneriti oppure "chiari" identificati con i bit di valore 1 e 0 rispettivamente.

Analogamente si ricavano i valori dei restanti 60 byte, la cui successione, com'è intuibile, procede da sinistra verso destra e dall'alto verso il basso.

Questa operazione può sembrare alquanto lunga e noiosa ma, in genere, si incontreranno molti byte vuoti (=Ø) o pieni (=255) ed alcuni con la stessa configurazione di bit.

I puntatori

Una volta calcolati i 63 valori da assegnare ad altrettanti byte consecutivi, bisogna trovare un'area di memoria esente da influenze esterne (ossia non riservata al sistema operativo) che funga da "contenitore" per questi byte.

Come vedremo meglio in seguito, l'indirizzo iniziale di quest'area deve essere un multiplo della costante 64 (ossia 0, 64, 128, 192, 256, e così via). Un'altra importante restrizione da imporre alla zona da ricercare consiste nel non oltrepassare l'indirizzo 16383, ma cambiando banco

di memoria si può facilmente superare questo ostacolo. A tal proposito si consiglia l'articolo: "La gestione dei quattro banchi di memoria" pubblicato sul N.32 di Commodore Computer Club.

Procedendo nella ricerca attraverso la mappa di memoria, troviamo due aree che soddisfano le regole sopra enunciate; la prima va da 704 a 767 e può contenere un solo sprite, mentre la seconda si estende da 832 a 1023 e può contenere ben tre sprite. Quest'ultima area non è libera nel senso stretto della parola, ma è adibita a buffer temporaneo del registratore. E' obbligatorio, pertanto, risistemarvi i dati riguardanti gli sprite ogniqualvolta si fa uso di questa periferica dal momento che vengono cancellati ogni volta che si utilizza il registratore.

In realtà ciò non costituisce un vero problema in quanto una volta letto il programma che gestisce gli sprite, il registratore non viene, in genere, adoperato per altre operazioni. Chi possiede il drive per dischetti, ovviamente, non ha di questi problemi (ci mancherebbe altro!).

Qualora dovesse rendersi necessario l'uso del registratore, è consigliabile trasferire prima i dati degli sprite in un'area protetta (va benissimo quella da 49152 a 53247), poi utilizzare la periferica e, infine, ritrasferire l'immagine codificata nelle locazioni originali.

Come si è visto, abbiamo trovato spazio sufficiente per soli quattro sprite. Sarebbe possibile riservare spazio a partire dalla locazione 2048 (agendo su alcuni puntatori) ma il discorso, oltre ad essere alquanto impegnativo e fuori argomento, è stato trattato esaurientemente nell'articolo dedicato alla grafica pubblicato sul numero 30 di Commodore Computer Club.

Ma il nostro lavoro non è ancora concluso: infatti non basta memorizzare da qualche parte i dati riguardanti gli sprite, ma bisogna indicare al VIC II DOVE sono situati questi ultimi.

Per ogni sprite esiste una locazione, compresa tra 2040 e 2047, che lo indirizza, fungendo da puntatore; allo sprite 0 è abbinata la locazione 2040, allo sprite 1 la 2041, fino allo sprite 7 che ha il suo puntatore in 2047.

Il valore **P** da assegnare a ciascun puntatore è dato dalla formula:

P=I/64

in cui I è la locazione contenente il primo byte dello sprite considerato e a questo punto si capisce perchè I deve essere un multiplo di 64.

Ricorriamo ad un esempio: supponiamo di aver memorizzato i 63 byte dello sprite N.3 a partire dalla locazione di indirizzo 960. Il valore da depositare in 2043 corrisponde a 15 (960/64=15) e, pertanto, dovremo eseguire:

POKE2043,15

In definitiva se cerchiamo di memorizzare i 63 byte di uno sprite a partire da un indirizzo non multiplo di 64, rischiamo di vederne apparire solo una parte, dato che il VIC II "ragiona", per ciò che concerne gli sprite, a blocchi di 64 byte.

Il registro di abilitazione

Questo registro (il registro è una locazione di memoria mappata in un chip di I/O come il VIC II) è locato in 53269 ed è chiamato Sprite Enable Register (registro di abilitazione sprite).

Come tutte le 65536 locazioni del C-64, anche questa è composta da otto bit. Immaginarne la funzione è estremamente semplice: ogni bit è abbinato allo sprite il cui numero d'ordine è equivalente alla posizione, o peso, del bit stesso.
ossia quello più a sinistra, rappresenta lo sprite 7 e così via.

Ponendo a 1 un certo bit di questa locazione, si dà al VIC II il "permesso" di visualizzare sullo schermo lo sprite implicato.

La traduzione in soldoni di quanto sopra descritto è presto fatta; bisogna eseguire:

POKE 53269, PEEK (53269) OR (2↑S7

dove S rappresenta il numero dello sprite di cui si desidera la visualizzazione.

Viceversa, per "spegnere" un certo sprite, sarà necessario digitare:

POKE 53269, PEEK (53269) AND (255-2↑S)

in cui S assume il solito significato.

I colori

Una volta disegnato, tradotto in byte, memorizzato, localizzato e visualizzato, uno sprite ha bisogno di una veste cromatica che gli permetta di risaltare o di mimetizzarsi con lo sfondo, a seconda delle situazioni.

Come per la grafica in alta risoluzione, vi sono due modi per visualizzare i colori di uno sprite: normale e multicolor.

Per stabilire che un determinato sprite dovrà essere visualizzato in modo normale, sarà sufficiente porre a zero il bit corrispondente (la relazione tra bit e sprite è la stessa di quella valente nel registro di abilitazione) mediante la seguente istruzione:

POKE 53276, PEEK (53276) AND (255-2↑S7

dove S assume ancora il ruolo di numero sprite (d'ora in avanti non sarà più ripetuta questa precisazione).

Una volta posto uno sprite in modo normale, bisogna assegnargli un colore tra i sedici possibili.

Il codice del colore scelto (vedere l'appendice relativa nel manuale del computer) dovrà essere depositato nella locazione il cui indirizzo è dato da 53287+S eseguendo:

POKE 53287+S,CC

dove CC è il colore da assegnare allo sprite S.

Il colore depositato in queste locazioni sarà assunto dalle celle del disegno che avevamo annerito (ossia i bit posti a 1); le celle della griglia rimaste chiare avranno il colore dello sfondo.

Se si desidera attivare il modo multicolor per un determinato sprite, dovremo eseguire:

POKE 53276, PEEK (53276) OR (2↑S7

Per quanto concerne questo modo di visualizzazione, daremo per scontato che chi legge queste righe conosca sufficientemente il suo funzionamento in generale. Si è preferito tralasciare questi dettagli in quanto si sarebbe dovuto rifare il discorso riguardante il disegno dello sprite, appesantendo molto l'articolo, peraltro già sostanzioso.

A beneficio di coloro che rientrano nella categoria sopra definita, facciamo notare che tutto quanto viene esposto è valido anche per il modo multicolor, ad eccezione, naturalmente, di ciò che segue in questo paragrafo.

I quattro possibili colori dello sprite abbinati alle coppie di bit 00, 01, 10, 11, sono dati, rispettivamente, dallo sfondo, dal registro multicolor 1 (53285), dal registro colore (53287+S) e dal registro multicolor 2 (53286).

Si noti che i due registri multicolor sono uguali per tutti gli otto sprite.

Le espansioni dimensionali

Un'altra delle già numerose opzioni è quella che consente di ingrandire un determinato sprite in senso verticale, orizzontale e in entranbi i sensi.

L'espansione avviene raddoppiando le dimensioni di ciascun pixel senza, per questo, raddoppiare la risoluzione.

Vi sono due registri, adibiti allo scopo, locati in 53271 e 53277: il primo permette l'espansione dello sprite in verticale, il secondo in orizzontale.

Come per altri casi già visti, si ottiene l'espansione verticale dello sprite S mediante la seguente istruzione:

POKE 53271, PEEK (53271) OR (2↑S7

mentre l'espansione orizzontale avviene eseguendo:

POKE 53277, PEEK (53277) OR (2↑S7

Viceversa, per ottenere la normalizzazione verticale e orizzontale, bisogna eseguire, rispettivamente:
POKE 53271, PEEK (53271) AND (255-2↑ S7
e
POKE 53277, PEEK (53277) AND (255-2↑ S7

La priorità

Prima di passare alla fase conclusiva, l'ultimo argomento da trattare è quello della priorità dello sprite rispetto allo sfondo.

Dare una priorità maggiore ad uno sprite, significa che, ogniqualvolta esso verrà posizionato (vedremo fra poco come) nella stessa area grafica occupata, per esempio, dal carattere "A", lo sprite apparirà "sopra" la "A", ossia la nasconderà.

Viceversa, assegnando una priorità maggiore allo sfondo, lo stesso sprite verrà nascosto dal carattere, facendo sembrare che sia quest'ultimo a stargli "davanti" o "sopra", come dir si voglia.
Per dare allo sprite S una priorità più bassa, basterà digitare:

POKE 53275, PEEK (53275) OR (2↑ S7
mentre, per ottenere l'effetto contrario, dovremo eseguire:
POKE 53275, PEEK (53275) AND (255-2↑ S7
Oltre alla priorità tra sprite e sfondo, esiste anche tra sprite e sprite un grado di priorità. Quest'ultima, come già detto, non è modificabile dall'utente ma si basa sul numero d'ordine degli sprite in causa: lo sprite che ha numero d'ordine più basso avrà priorità più alta e nasconderà l'altro sprite.

Di conseguenza lo sprite che, in assoluto, sarà visualizzato sempre sopra tutti gli altri, sarà il numero 0; viceversa il numero 7 sarà sempre nascosto dai rimanenti.

Il posizionamento sullo schermo

Lo schermo che l'integrato VIC II costruisce è composto da 320 pixel orizzontali per 200 verticali.

Uno sprite può assumere qualsiasi posizione all'interno di quest'area e anche, come vedremo, all'esterno di essa.

Prima di continuare nel discorso, precisiamo che lo schermo è considerato dal VIC II come un piano le cui dimensioni e posizioni sono quantizzate da uno strano sistema di riferimento cartesiano ortogonale avente origine nel punto in alto a sinistra e i cui assi sono finiti e discreti, ossia accettano come coordinate solo interi positivi varianti, nel caso delle ascisse, da 0 a 319 e, nel caso delle ordinate, da 0 a 199.

Un'altra particolarità di questo sistema (la si può estrapolare dalle precedenti indicazioni) consiste nel fatto che, in pratica, si ha a disposizione un solc quadrante il cui asse Y è positivo verso il basso.

Una volta completate tutte le premesse teoriche possiamo, finalmente, vedere come si posiziona uno sprite sullo schermo.

Per definire la posizione di uno sprite abbiamo bisogno di due numeri, di cui il primo rappresenta la posizione orizzontale (ascissa) e l'altro la posizione verticale (ordinata).

Incominciamo con quest'ultima. Ogni sprite, com'era prevedibile, ha delle locazioni di memoria che indicano le coordinate alle quali esso viene visualizzato.

La locazione LY che funge da deposito per l'ordinata di uno sprite S è data dall'espressione:
LY = 53249+(2*S)

Ad esempio, l'ordinata dello sprite 5 è depositata nella locazione 53259 (=53249+(2*5)). Variando il valore di questa locazione lo sprite, se è stato precedentemente abilitato, si muoverà automaticamente lungo l'asse Y.

Incrementando o decrementando periodicamente e con una certa velocità una o entrambe le coordinate, si avrà l'impressione che lo sprite si muova con moto continuo.

Per stabilire la coordinata Y di un certo sprite S è sufficiente eseguire:
POKE 53249+(2*S),Y.

Naturalmente, una volta stabilite le costanti S ed Y la POKE risulterà meno complessa.

Prima di procedere con l'esame della coordinata X, dovremo fare una importantissima considerazione: come sappiamo, lo schermo ha 200 pixel verticali, ma la Y di uno sprite può valere da 0 a 255; come si conciliano queste due cose?

Incominciamo col dire che le coordinate di uno sprite si riferiscono al suo pixel in alto a sinistra.

A questo punto l'arcano è presto spiegato: assegnando alla coordinata Y valori troppo alti o troppo bassi lo sprite si "nasconde", parzialmente o per intero, sotto i bordi superiore o inferiore.

Per uno sprite non espanso verticalmente, il valore minimo da assegnare ad Y perchè si veda almeno la sua ultima riga è 30. Ciò vuol dire che dando ad Y il valore, per esempio, 28 lo sprite è totalmente nascosto dal bordo superiore.

Il massimo valore che si può attribuire ad uno sprite non espanso affinchè si veda per intero è 229. Al di sopra di questo limite, lo sprite comincerà a scomparire sotto il bordo inferiore dello schermo.

Il discorso, per quanto riguarda il posizionamento orizzontale, si fa leggermente più complesso in quanto implica due locazioni di memoria. Questo è dovuto al fatto che con gli otto bit contenuti in un byte si possono rappresentare 256 (2↑8) valori, mentre il valore massimo dell'ascissa di schermo è 319.
Per ovviare a questo secondo inconveniente è stata approntata un'altra locazione che contiene il nono bit (virtuale) della posizione X di ognuno degli otto sprite.

```
10 FOR I = 832 TO 959: POKE I,255: NEXT
20 POKE 2040,13: POKE 2041,14
30 POKE 53271,0: POKE 53277,0: POKE 53276,0: POKE 53264,0
40 POKE 53287,1: POKE 53288,0: POKE 53280,0: POKE 53281,12
50 POKE 53248,150: POKE 53249,130
60 POKE 53250,190: POKE 53251,130: POKE 53269,3
```

Figura 2: Listato BASIC che abilita gli sprite 0 e 1 e li pone al centro dello schermo.

In questo modo si avranno addirittura 512 (2 ↑ 9) possibili posizioni orizzontali per ogni singolo sprite. La locazione 53264 è adibita a contenere gli otto "noni" bit associati ai corrispondenti sprite.

Senza entrare in ulteriori dettagli tecnici, diremo solo che, per assegnare un valore X allo sprite S, variante da 0 a 511, basta digitare le seguenti istruzioni in un'unica linea di comandi Basic:

AX%=X/256:POKE53248+(2*S),X-(AX%*256):

POKE53264,PEEK(53264)ORAX%*(2 ↑ S7.

E' una linea piuttosto lunga, ma sostituendo le costanti X ed S e ricalcolando le espressioni interne si accorcerà di molto.

Anche in questo caso, le X assegnabili superano di gran lunga le coordinate di schermo orizzontali. Assegneremo, pertanto, alcuni dei limiti ai quali deve sottostare l'ascissa di uno sprite affinchè sia visualizzato in parte o per intero.

Per essere almeno in parte visibile, la coordinata X deve risiedere nell' intervallo chiuso di valori che vanno da 1 a 343 (parliamo di sprite non espansi).

I dati che abbiamo fornito, riguardanti i limiti di visualizzazione, non sono completi nè, per ovvie ragioni di spazio, potrebbero esserlo. Ma questo deve essere uno sprone per i lettori che desiderano saperne di più.

Per incentivare maggiormente gli interessati nella ricerca di questi limiti, diamo qui di seguito alcuni consigli:

1) digitare ed eseguire il programmino di figura 3;

2) se tutto procede bene, dovrebbero apparire due sprite a forma di rettangolo pieno al centro dello schermo: quello bianco è il numero 0, quello nero è il numero 1;

3) per trovare i valori limite per i quali uno sprite si vede, non si vede o si vede parzialmente, muovete uno dei due sprite mediante le linee sopra indicate digitate in modo diretto, prendendo nota dei valori trovati;

4) espandete, sempre per mezzo di istruzioni in modo diretto, lo sprite nei diversi modi possibili e ripetete le operazioni descritte nel punto 3.

5) Per sperimentare le diverse priorità sprite-sprite e sprite-sfondo, muovete entrambe gli sprite uno sopra (sotto) l'altro e sopra (sotto) caratteri digitati precedentemente; ancora una volta è consigliabile

digitare in modo diretto le istruzioni relative alle varie azioni.

Gli scontri (incontri) degli sprite

Abbiamo riservato l'ultimo posto a questo argomento in quanto l'utilizzazione di questa "performance" per mezzo del Basic è alquanto difficoltosa e rallenta ulteriormente la (già di per sè) lenta gestione degli sprite.

Purtuttavia, ci siamo sentiti il dovere di trattare questo argomento in quanto, oltre a completare quella che è voluta essere una descrizione generale del funzionamento degli sprite, potrà essere oggetto di "libidine" soprattutto per coloro che conoscono il linguaggio macchina del C/64.

Ciò non vuol dire che col Basic non si può fare nulla, tanto è vero che la trattazione che ci apprestiamo a fare sarà concretizzata mediante linee Basic perfettamente funzionanti.

Dovremo, prima di procedere, distinguere due tipi di collisioni: tra sprite diversi e tra sprite e sfondo.

Vi è un registro (Sprite to sprite collision detect register) locato in 53278 che registra le collisioni avvenute tra gli sprite.

Gli otto bit di questo registro sono abbinati agli otto sprite secondo la solita relazione. Normalmente questi bit sono posti a zero; allorquando due sprite vengono a sovrapporsi, anche in parte, i bit associati vengono posti ad uno.

Tentando di "leggere" questo registro, con l'istruzione PRINT PEEK (53278) o con altri sistemi, i bit posti ad uno si azzereranno automaticamente.

Dall'analisi del valore decimale ricavato dalla PEEK si otterrà un responso non univoco in quanto se, per esempio, due coppie di sprite si sono scontrate contemporaneamente, si avranno quattro bit accesi e non si potrà stabilire quali sono stati i contatti.

A questo inconveniente si può ovviare andando a leggere le posizioni dei vari sprite coinvolti e confrontandole tra di loro.

Volendo sapere, durante l'elaborazione di un programma, se lo sprite S1 e lo sprite S2 si sono sovrapposti, si può utilizzare la linea:

SS=PEEK(53278):

IFSS=(2 ↑ S17842 ↑ S27THEN linea azione

La "linea azione" è quella particolare linea alla quale si vuole che il programma salti se i due sprite sono entrati in collisione.

Questa linea (e le sucessive) potrebbe contenere, per esempio, le istruzioni che simulano il rumore di un'esplosione, nel caso si tratti di un videogioco nel quale un proiettile colpisca un bersaglio, e via dicendo.

L'altro tipo di collisione, cioè quello tra sprite e sfondo, è rivelato dal registro di indirizzo 53279 (Sprite to background collision detect register).

Anche in questo registro gli otto bit sono associati agli otto sprite; il funzionamento è analogo a quello del registro precedente. Basta, infatti, che un determinato sprite si scontri, ad esempio, con un carattere dello schermo per far accendere il relativo bit all'interno del registro.

Il rilevamento va effettuato con la seguente linea:

SB=PEEK(53279):

IFSB=2 ↑ S THEN linea azione.

Come si può notare, lo sprite S coinvolto è uno solo: per individuare collisioni multiple bisogna utilizzare espressioni molto complesse e, pertanto, lente; è a questo punto che l'uso del linguaggio macchina diventa quasi obbligatorio.

```
  1 REM  PROGRAMMA DIMOSTRATIVO
  2 REM  BY PASQUALE D' ANDRETI
  3 REM  RIARDO  (CASERTA)
  6 :
 10 POKE 53280,0:POKE 53281,0
 20 GOSUB 700
 30 HX=53248:HY=HX+1:CX=HY+1:CY
    =CX+1:SS=53278:SC=0
 40 FOR I=896 TO 1022:READ A:PO
    KE I,A:NEXT
 50 POKE 2040,14:POKE 2041,15:P
    OKE 53271,0:POKE 53277,0:PO
    KE 53276,0:POKE 53248,0
 55 POKE 53250,0:POKE 53264,0:P
    OKE 53287,2:POKE 53288,1:PO
    KE 53269,3
 60 PRINT"[CLEAR]"
 80 XH=24:YH=50:POKE HX,XH:POKE
     HY,YH
 90 FOR V=0 TO 16
100 AX=INT(RND(.)*232)+23:AY=IN
    T(RND(.)*179)+51
110 POKE CX,AX:POKE CY,AY:TI$="
    000000"
120 L=PEEK(197):IF L=41 AND YH>
    54 THEN YH=YH-4
130 IF L=44 AND YH<228 THEN YH=
    YH+4
140 IF L=42 AND XH>28 THEN XH=X
    H-4
150 IF L=50 AND XH<252 THEN XH=
    XH+4
160 POKE HX,XH:POKE HY,YH
180 IF PEEK(SS)=3 THEN SC=SC+1
185 PRINT"[HOME]"SC
190 IF TI>200 THEN NEXT:GOTO 21
    0
200 GOTO 120
210 PRINT"[CLEAR][GRIGIO3]
      HAI REALIZZATO "SC"  PUNT
    I"
220 PRINT"[17 DOWN]VUOI GIOCARE
     ANCORA (S/N)?"
230 POKE 53271,3:POKE 53277,3
240 POKE HX,180:POKE HY,105
250 POKE CX,135:POKE CY,100
260 FOR T=1 TO 1000
270 POKE 198,0:WAIT 198,1:POKE
    53269,0:GET A$:IF A$="S" TH
    EN  RUN
280 PRINT"[CLEAR]":END
700 PRINTCHR$(142);CHR$(8)
710 PRINT"[CLEAR][GRIGIO3]
      IL GIOCO DELLA RINCORSA.
    "
720 PRINT"[3 DOWN]CERCA DI RAGG
    IUNGERE  IL  SIMBOLO  DELLA
730 PRINT"COMMODORE MENIRE SI S
    POSTA.
740 PRINT"[DOWN]PER MUOVERTI US
    A I TASTI:
745 PRINT"[DOWN]
    SU'
750 PRINT"[DOWN]
     P
760 PRINT" SINISTRA     L      ;
       DESTRA
770 PRINT"                      .
780 PRINT"[DOWN]
    GIU'
790 PRINT"[DOWN]IL PUNTEGGIO SA
    RA' VISUALIZZATO IN  ALTO
800 PRINT"A SINISTRA.
810 PRINT"[2 DOWN][RVS] PREMI U
    N TASTO PER INCOMINCIARE"
820 POKE 198,0:WAIT 198,1:RETUR
    N
900 DATA 0,0,0,1,128,192,3,227,
    224,7,247,240,7,255,240,7,2
    55,240
910 DATA 3,255,224,3,255,224,1,
    255,192,1,255,192,0,255,128
920 DATA 0,127,0,0,62,0,0,28,0,
    0,8,0,0,0
930 DATA 0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0
940 DATA 0,120,0,3,254,0,15,254
    ,0,31,254,0,63,134,0,126,1,
    254,124,1
950 DATA 252,120,1,248,248,1,24
    0,248,1,224,248,0,0,248,1,2
    24,248,1,16,120
960 DATA 1,8,124,1,4,126,1,254,
    63,134,0,31,254,0,15,254,0,
    3,254,0,0,120,0
```

Da oggi c'è un nuovo distributore di stampanti FACIT per il tuo Personal Computer IBM

Agenzie FACIT

Arenzano (GE) P.za degli Ulivi, 15 - Tel.: 010/9112036
Bergamo *D.I.P. Bergamo* Via Borgo Palazzo, 90
Tel.: 035/233909
Bologna *D.I.P. Bologna* P.za Porta Mascarella, 7
Tel.: 051/240602
Castelfranco Veneto (TV) *Vecom* Borgo Treviso, 45
Tel.: 0423/496222
Fabriano (AN) *D.I.P. Ancona* Via G. Tommasi, 15
Tel.: 0732/22259
Livorno *D.I.P. Livorno* Via Alfieri, 19
Tel. 0586/422377
Milano *D.I.P. Milano* Via A. Costa, 33
Tel.: 02/2840508-2840488
Roma *D.I.P. Roma* Via C. Colombo, 179
Tel.: 06/5133041
San Mauro Torinese (TO) *Elcomin* Corso Lombardia, 75
Autoporto Pescarito - Tel.: 011/2735501-2-3

Distributori FACIT

Bassano del Grappa (VI) *Studio L. & C. SpA* V.le Diaz, 27 - Tel. 0424/212541
Belluno *SCP Computer System* Via Feltre, 244
Tel. 0437/20826
Castelfranco V. (TV) *Volpato snc* Via Riccati, 25
Tel. 0423/495961
Gorizia *Quark srl* Via Udine, 143 Tel. 0481/391693
Mestre Loc. Chirignago
Computime srl
Via Miranese, 420
Tel. 041/917566
Mestre/Venezia
Bit Computers srl
P.za Barche, 45
Tel. 041/958007

Mestre/Venezia *Boffelli El. Servizi srl* - C.so del Popolo, 32 - Tel. 041/5053333
Montebelluna (TV) *Volpato snc* Via Montegrappa, 103
Tel. 0423/302771
Padova *System Ros sas* P.za De Gasperi, 14
Tel. 049/38412
Pordenone *Strutture Informatiche srl* Via S. Caterina, 3
Roma *Data Office* Via Sicilia, 205 - Tel. 06/4742651
Roma *Expo* Via IV Novembre - Tel. 06/6783488
Roma *Valde Adel* P.za Bainsizza, 3
Tel. 06/316331-316676
S. Donà di Piave *Computime srl* Via Vizzotto, 13
Tel. 0421/44505
Schio (VI) *Bit srl* Via Roccoletto, 23
Tel. 0445/28928
Schio (VI) *Linea 4 snc* Via Riva del Cristo, 4/8
Tel. 0445/28970
Tavernelle (VI) *Centro Informatica srl* Via Verona, 64
Tel. 0444/573967
Treviso *Informatica Tre srl* V.le della Repubblica, 19/B
Tel. 0422/65993
Trieste *Murri snc* Via A. Diaz, 24/A - Tel. 040/306091
Udine *GC Michieli snc* V.le Ungheria, 64
Tel. 0432/291835
Verona *Computek Sistemi srl* V.le del Lavoro, 33
Tel. 045/509311
Vicenza *Centro Informatica srl* C.so Fogazzaro, 28
Tel. 0444/38513

Centro Direz. Colleoni
Palazzo Orione Ingr. 1
20041 Agrate Brianza (MI)
Tel.: 039/636331
Telex 326423 SIAV BC

omaggio veramente favoloso!

Senza alcun tuo impegno, compila in ogni sua parte il tagliando e consegnalo a un distributore **FACIT**

Cognome ....

Nome .... Età ....

Indirizzo ....

....

Città .... C.a.p. ....

Professione ....

Eventuale computer in tuo possesso

.... CO

Character editor per C/128 in 80 colonne

Come ridefinire i caratteri in modo C/128 su video a 80 colonne

di Giorgio Chiozzi

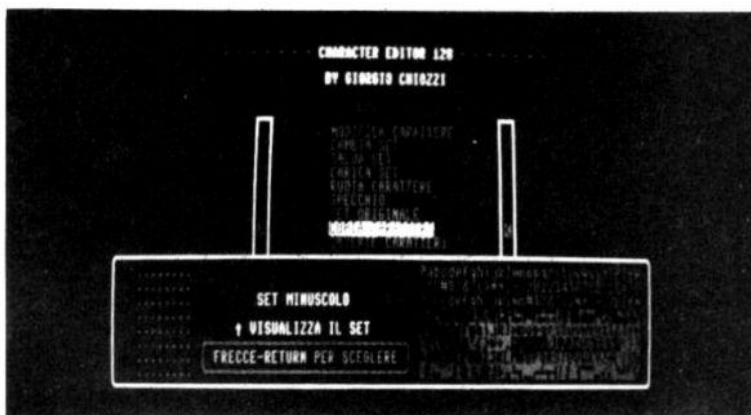
Il processore video del Commodore 128, l'8563, può visualizzare una matrice di caratteri di 25 righe per 80 colonne, utilizzando come memoria immagine una parte del banco Dram, a partire dall'indirizzo $2000, con un passo di 16 byte per carattere, ed un "interspazio" di 8 byte inutilizzati.

Ogni volta che si accende il computer, il set di caratteri deve essere caricato in Dram dalla Rom in cui il set è memorizzato a partire dall'indirizzo $D000.

E' quindi particolarmente semplice modificare uno o più caratteri, essendo sufficiente modificare i corrispondenti byte Dram. Il programma "Character Editor C/128", scritto quasi interamente in Basic, vuole essere un esempio di come può venire affrontato il problema della definizione dei caratteri sullo schermo ad 80 colonne del C/128. Anche se tutto in teoria può apparire semplicissimo, in realtà bisogna fare i conti con le già lamentate "carenze" hardware della macchina, quali l'impossibilità di accesso diretto al banco Dram.

Il programma utilizza una zona Ram buffer, allocata all'indirizzo $E000, in banco 0, come memoria di lavoro per i caratteri ridefiniti. Questa può venire trasferita nella reale mappa caratteri in Dram, per controllare l'effetto che i nuovi caratteri provocano sullo schermo ad 80 colonne. Ciò si rende indispensabile qualora si ridefiniscano caratteri speciali per giochi o programmi grafici, caratteri che renderebbero molto fastidioso l'utilizzo di Character editor, che lavora sullo schermo ad 80 colonne.

Il programma fa largo uso di semplicissime routine in linguaggio macchina: a partire dall'indirizzo decimale 3633 è allocata la routine che ricopia il set ridefinito dal buffer nella reale zona di memoria carattere Dram.

All'indirizzo decimale 3584 è allocata la routine che riempie la Ram buffer con i caratteri residenti su Rom; essa è usata per inizializzare il buffer o per cancellare il set precedentemente definito.

All'indirizzo 52684 è allocata la routine del sistema operativo che esegue l'update (=modifica) di un registro del processore video, ed all'indirizzo 52696 quella che ne esegue la lettura. Infine, all'indirizzo 52748, è allocata la routine che carica in Dram i caratteri memorizzati in Rom.

Da notare le linee che eseguono rispettivamente le operazioni di poke e peek sui byte Dram della memoria dedicata al processore video.

L'uso del programma

• Startup: dato il comando Run, il programma chiede se deve essere inizializzato il buffer, ossia l'area di lavoro per i nuovi caratteri. Rispondere "Y" se in memoria non vi sono caratteri ridefiniti su cui si sta lavorando, altrimenti "N".

• Menu: il menu è composto da 9 opzioni di lavoro, che possono essere selezionate usando i tasto Crsr in combinazione con i tasti Shift e Return.

• 1 Edit char: il carattere da modificare, o da creare ex novo, viene selezionato usando i tasti Crsr; il tasto Return convalida la scelta, il tasto spazio consente il ritorno al menu. Scelto il carattere, questo viene visualizzato ingrandito sulla sinistra: i tasti Crsr muovono il cursore, lo spazio inverte i pixel, Clr/Home cancella il carattere, freccia in alto shifta il carattere in alto, meno (-) lo shifta a sinistra; con Return si esce. Per convalidare il nuovo carattere premere "Y" oppure Return.

• 2 Change char set: cambia il set di caratteri su cui lavorare da maiuscolo a minuscolo o viceversa.

• 3 Save char set: salva su disco il buffer in cui sono memorizzati i nuovi caratteri. E' possibile salvare solo il set maiuscolo o solo quello minuscolo, rispondendo "Y" alla domanda: "Only Upper (o Lower) set?". Se si desidera salvare tutti i caratteri, premere "N". Il tasto Return fa tornare al menu.

Il programma aggiunge automaticamente al nome impostato il prefisso "CHR-(" ed il suffisso ")". Pertanto un set di caratteri memorizzato impostando "PIPPO" sarà presente, sulla directory, con "CHR-(PIPPO)".

• 4 Load char set: carica da disco un set di caratteri specificato indicando il nome del set (privo del suffisso e prefisso visti prima).

• 5 Rotate char: consente di ruotare i caratteri di 90 gradi in senso orario; la scelta del carattere avviene in modo analogo a quella dell'opzione "Edit Char", quindi il carattere viene visualizzato ingrandito: un tasto esegue la rotazione, Return esce. Per convalidare il carattere ruotato premere "Y" oppure Return.

• 6 Char mirror: esegue il simmetrico dei caratteri rispetto all'asse x oppure y; scelto il carattere, "H" ne esegue il simmetrico rispetto all'asse y, "V" rispetto all'asse x. Return esce, "Y" oppure Return convalidano il carattere.

• 7 Original char set: copia in buffer i caratteri contenuti su Rom, cancellando quelli eventualmenti ridefiniti dall'utente; alla domanda "Are you sure?" rispondere "Y" se si desidera realmente ripristinare i caratteri originali.

• 8 Copy char block: copia un blocco di caratteri: è necessario scegliere il primo e l'ultimo carattere del blocco "origine", scegliere se si desidera copiare il blocco all'interno dello stesso set di caratteri, maiuscolo o minuscolo, ed infine scegliere il punto di applicazione della copia. Il tasto spazio torna al menu.

• 9 Reverse char block: inverte i pixels di un blocco di caratteri; è necessario scegliere il primo e l'ultimo carattere di tale blocco. con il tasto spazio si torna al menu.

I lettori più attenti avranno sicuramente notato che il blocco di Ram che viene utilizzato come buffer, e che occupa 4 kbyte, è allocato all'indirizzo $E000, quando avrebbe potuto essere allocato all'indirizzo $F000; ciò in quanto il programma è stato sviluppato con l'estensione Basic "Graphic Expander C/128", che aggiunge, tra gli altri, i comandi hpoke ed hpeek che, consentendo la manipolazione diretta da Basic del banco Dram, evitano di dover ricorrere alle lente procedure di accesso alla memoria video e rendono molto più veloce il programma stesso.


```
1000 REM
1010 REM              CHARACTER EDITOR 128 BY GIORGIO CHIOZZI
1020 REM
1030 :
1040 REM CARICAMENTO ROUTINE LINGUAGGIO MACCHINA
1050 FAST:FORI=0TO123:READA:POKEI+3584,A:NEXT:FORI=0TO3:READCH(I):NEXT
1060 :
1070 REM DEFINIZIONE LIMITE MEMORIA BASIC
1080 POKE4627,239:POKE4626,255
1090 :
1100 REM DEFINIZIONE CARATTERI SPECIALI
1110 Q$=CHR$(17):C$=CHR$(159):B$=CHR$(5):D$=CHR$(29):G$=CHR$(30):ES$=CHR$(27)
1120 GI$=CHR$(158):SO$=CHR$(2):LA$=CHR$(15):L$=CHR$(192):HO$=CHR$(19)
1130 UP$=CHR$(145):LG$=CHR$(153):FORI=1TO28:WD$=WD$+L$:NEXTI:CH$=CHR$(147)
1140 WB$=CHR$(202)+WD$+CHR$(203):WD$=CHR$(213)+WD$+CHR$(201):SI$=CHR$(157)
1150 RE$=CHR$(28):RV$=CHR$(18):RO$=CHR$(146):FORI=1TO28:BK$=BK$+CHR$(32):NEXTI
1160 FORI=1TO19:M$=M$+L$:NEXTI:FORI=1TO14:QC$=QC$+Q$:NEXTI
1170 W$=D$+CHR$(213)+M$+CHR$(177)+CHR$(192)+CHR$(177)+LEFT$(M$+M$,32)
1180 W$=W$+CHR$(177)+CHR$(192)+CHR$(177)+M$+CHR$(201)
1190 FORI=1TO76:WW$=WW$+L$:NEXTI:WW$=D$+CHR$(202)+WW$+CHR$(203)
1200 :
1210 REM CODICE ROUTINE LINGUAGGIO MACCHINA
1220 :
1230 DATA169,0,160,208,162,224,133,218,133,170
1240 DATA132,219,134,171,160,0,162,14,169,0
1250 DATA141,0,255,169,218,32,116,255,162,63
1260 DATA142,0,255,145,170,200,208,234,230,171
1270 DATA230,219,165,171,201,240,144,224,96,169
1280 DATA0,160,224,133,170,132,171,169,0,141
1290 DATA0,255,162,18,169,32,32,204,205,232
```

```
1300 DATA169,0,32,204,205,160,0,162,63,142
1310 DATA0,255,177,170,162,0,142,0,255,32
1320 DATA202,205,152,41,7,73,7,240,12,200
1330 DATA208,231,230,171,165,171,201,240,144,223,96
1340 DATA162,8,134,102,32,202,205,198,102,208,249,240,231
1350 DATA64,32,96,160:TRAP3110
1360 BANK15:FAST:SYS52748:SCNCLR
1370 PRINTTAB(16)C$"- - - - - - - "B$"CHARACTER EDITOR 128"C$" - - - - - - - -"
1380 PRINTTAB(31)Q$B$"BY GIORGIO CHIOZZI":TC-35:WT-81
1390 PRINTTAB(35)Q$LA$SO$C$"MENU'":BW-1285:OT-80:OK-80*7:UN-1:TE-7
1400 HI-18:LO-19:SY-52684:SK-SY-2:DD-255:DT-256:RE-52696:SI-6:RO-95:BG-3373
1410 PRINTTAB(21)CHR$(142)B$CHR$(176)CHR$(192)CHR$(174);
1420 PRINTTAB(56)CHR$(176)CHR$(192)CHR$(174)
1430 FORI-1TO9:READA$(I):PRINTTAB(21)B$CHR$(221)CHR$(32)CHR$(221);
1440 PRINTG$TAB(32)A$(I)TAB(56)B$CHR$(221)CHR$(32)CHR$(221)
1450 NEXTI:K-1:SE-0:FORI-0TO7:U(I)-2↑I:NEXT:KI-0
1460 PRINTW$:FORI-1TO8:PRINTB$D$CHR$(221)TAB(78)CHR$(221):NEXTI:PRINTWW$;
1470 GOSUB2980:GOSUB3010
1480 PRINTUP$UP$UP$UP$:PRINTTAB(14)LG$WD$
1490 PRINTTAB(14)CHR$(221)TAB(43)CHR$(221):PRINTTAB(14)WB$
1500 CHAR1,22,18,B$+"SET MAIUSCOLO":CHAR1,19,20,"↑ VISUALIZZA IL SET"
1510 DATA MODIFICA CARATTERE,CAMBIA SET,SALVA SET,CARICA SET,RUOTA CARATTERE
1520 DATA SPECCHIO,SET ORIGINALE,COPIA CARATTERI,INVERTE CARATTERI
1530 CHAR1,23,22,LA$+"INIZIALIZZO?"+ES$+"O"
1540 GETKEYA$:IFA$-"S"THENSYS3584:GOTO1560
1550 IFA$<>"N"THEN1540
1560 CHAR1,16,22,LA$+GI$+"FRECCE-RETURN PER SCEGLERE"+ES$+"O"
1570 IFKI-1THENSYS3633
1580 PRINTHO$TAB(22)LEFT$(QC$,5+K)RE$RV$">"TAB(32)B$A$(K)TAB(57)RE$RD$"<"
1590 GETKEYA$:PRINTHO$TAB(22)LEFT$(QC$,5+K)" "TAB(32)A$(K)TAB(57)" "
1600 IFA$-Q$THENK-K+1+9*(K-9):GOTO1580
1610 IFA$-UP$THENK-K-1-9*(K-1):GOTO1580
1620 IFA$-"↑"THENKI-XOR(KI,1):SYS3633-49115*(KI-0)
1630 IFA$-CHR$(13)THENONKGOTO1860,2350,2200,2300,2550,2390,1670,2660,2870
1640 GOTO1580
1650 :
1660 REM CANCELLAZIONE SET CARATTERI RIDEFINITI
1670 CHAR1,15,22,BK$:CHAR1,23,22,LA$+B$+"SEI SICURO ?"+ES$+"O"
1680 GETKEYA$:IFA$-"S"THENSYS3584
1690 CHAR1,15,22,BK$:GOTO1560
1700 REM
1710 REM EDIT CHAR
1720 REM
1730 CHAR1,16,22,B$+LA$+"RETURN-SCEGLIE SPAZIO-ESCE"+ES$+"O"
1740 IFKI-1THENSYS3633
1750 W-BG+X+Y*OT:PO-W:GOSUB3080:BY-RORRO:GOSUB3050:GETKEYA$:BY-R:GOSUB3050
1760 IFA$-D$THENX-X+1+32*(X-31)
1770 IFA$-SI$THENX-X-1-32*(X-0)
1780 IFA$-UP$THENY-Y-1-8*(Y-0)
1790 IFA$-Q$THENY-Y+1+8*(Y-7)
1800 IFA$-" "THENCHAR1,15,22,BK$:RETURN
1810 IFA$-"↑"THENKI-XOR(KI,1):SYS3633-49115*(KI-0)
1820 IFA$<>CHR$(13)GOTO1750
1830 CHAR1,15,22,BK$:RETURN
1840 :
1850 REM MODIFICA/CREA CARATTERE
1860 GOSUB1730:IFA$-" "THEN1560
1870 CHAR1,15,22,B$+LA$+"SPAZIO INVERTE, RETURN FATTO"
1880 CHAR1,20,16,LG$+"↑ E ← PER SHIFTARE"+B$+ES$+"O"
1890 GOSUB2480:GOSUB1940:CHAR1,15,16,BK$:CHAR1,15,22,BK$:CHAR1,26,22,"OKAY ?"
1900 GETKEYA$:CHAR1,15,22,BK$:IFA$-CHR$(13)ORA$-"Y"THENBEGIN
```

```
1910 FORI=0TO7:W=0:FORII=0TO7:PO=BW+II+I*OT:GOSUB3080:W=W+U(7-II)*(-(R=81)):NEXT
1920 POKEU+I,W:NEXTI:BEND
1930 GOTO1860
1940 CX=0:CY=0
1950 W=3333+CX+CY*80:PO=W:GOSUB3080:BY=ROR95:GOSUB3050:GETKEYA$:BY=R:GOSUB3050
1960 IFA$=D$THENCX=CX+1+8*(CX=7)
1970 IFA$=SI$THENCX=CX-1-8*(CX=0)
1980 IFA$=UP$THENCY=CY-1-8*(CY=0)
1990 IFA$=Q$THENCY=CY+1+8*(CY=7)
2000 IFA$=" "THEN2100
2010 IFA$="↑"THENBEGIN
2020 FORI=0TO7:PO=BW+I:GOSUB3080:J=R:FORII=1TO7:PO=BW+I+II*OT:GOSUB3080
2030 PO=BW+I+(II-UN)*OT:BY=R:GOSUB3050:NEXT:PO=BW+I+OK:BY=J:GOSUB3050:NEXT:BEND
2040 IFA$="←"THENBEGIN
2050 FORI=0TO7:PO=BW+I*OT:GOSUB3080:J=R:FORII=1TO7:PO=BW+I*OT+II:GOSUB3080
2060 PO=PO-UN:BY=R:GOSUB3050:NEXT:PO=BW+I*OT+TE:BY=J:GOSUB3050:NEXT:BEND
2070 IFA$="コ"THENGOSUB3010
2080 IFA$<>CHR$(13)GOTO1950
2090 RETURN
2100 PO=W-2048:GOSUB3080:BY=81+35*(R=81):GOSUB3050:GOTO1950
2110 WINDOW15,22,42,22,1:PRINTTAB(3)"NOME FILE:";:L=0:N$=""
2120 PRINTCHR$(175)SI$;:GETKEYA$:PRINTCHR$(32)SI$;
2130 IFA$=CHR$(13)THENN$="CHR-("+N$+")":RETURN
2140 IFA$=CHR$(20)ANDL>0THENL=L-1:N$=LEFT$(N$,L):PRINTA$;
2150 IFA$>=" "ANDA$<="←"ANDL<10ANDA$<>CHR$(34)ANDA$<>","THENBEGIN
2160 PRINTA$;:N$=N$+A$:L=L+1:BEND
2170 GOTO2120
2180 :
2190 REM SAVE SET CARATTERI
2200 GOSUB2110:IFN$=""THEN2270
2210 PRINTCH$LA$TAB(4)"SOLO SET "MID$("MAIMIN",1-3*(SE=1),3)"USCOLO ?";
2220 GETKEYA$:IFA$<>"N"ANDA$<>"S"THEN2220
2230 PRINTCH$TAB(9)"SAVING ...";
2240 A=57344-2048*(SE=1ANDA$="S"):B=A+2048-2048*(A$="N")
2250 BSAVE(N$),D0,B0,P(A)TOP(B)
2260 PRINTCH$LA$TAB(2)DS$;
2270 GETKEYA$:PRINTCH$ES$HO$HO$:GOTO1560
2280 :
2290 REM LOAD SET CARATTERI
2300 GOSUB2110:IFN$=""THEN2270
2310 PRINTCH$TAB(9)"LOADING ...";
2320 BLOAD(N$),D0,B0:GOTO2260
2330 :
2340 REM SWITCH SET MAIUSCOLO-MINUSCOLO
2350 SE=XOR(SE,1):CHAR1,22,18,B$+"SET "+MID$("MAIMIN",1+3*SE,3)+"USCOLO"
2360 GOSUB2980:GOTO1560
2370 :
2380 REM CREAZIONE DEL SIMMETRICO DI UN CARATTERE
2390 GOSUB1730:IFA$=" "THEN1560
2400 GOSUB2480:CHAR1,17,22,LA$+"ORIZONTALE O VERTICALE?"+ES$+"O"
2410 CHAR1,15,16,LA$+C$+"PREMI 'O' O 'V', RETURN ESCE"+ES$+"O"
2420 GETKEYA$:IFA$=CHR$(13)THENCHAR1,15,22,BK$:CHAR1,15,16,BK$:GOTO2390
2430 IFA$<>"O"ANDA$<>"V"THEN2420
2440 CHAR1,15,22,BK$:CHAR1,20,22,LA$+"STO LAVORANDO ..."+ES$+"O"
2450 BANK0:FORI=0TO7:A(I)=PEEK(U+I):NEXT:BANK15
2460 IFA$="O"THEN2510
2470 FORI=0TO7:POKEU+I,A(7-I):NEXT:GOTO2400
2480 U=57344+(X+Y*32)*8+2048*SE:FORI=UTOU+7:BANK0:H=PEEK(I)
2490 FORII=0TO7:PO=BW+(I-U)*OT+II:BY=WT+(TC*((HAND(U(TE-II)))=ZE))
2500 BANK15:GOSUB3050:NEXTII,I:RETURN
2510 FORI=0TO7:D=0:FORII=0TO7:D=D-(U(II))*((A(I)AND(U(7-II)))>0):NEXTII
```

```
2520 POKEU+I,D:NEXTI:GOTO2400
2530 :
2540 REM RUOTA UN CARATTERE
2550 GOSUB1730:IFA$=" "THEN1560
2560 GOSUB2480:CHAR1,16,22,LA$+"UN TASTO RUOTA RETURN ESCE"+ES$+"O"
2570 GETKEYA$:IFA$=CHR$(13)THENCHAR1,15,22,BK$:GOTO2550
2580 CHAR1,15,22,BK$:CHAR1,20,22,LA$+"STO LAVORANDO ..."+ES$+"O"
2590 BANK0:FORI=0TO7:A(I)=PEEK(U+I):NEXT:BANK15
2600 FORI=0TO7:P(I)=0:FORII=0TO7
2610 P(I)=P(I)-U(II)*((A(II)ANDU(7-I))>0):NEXTII,I
2620 FORI=0TO7:POKEU+I,P(I):NEXT:GOTO2560
2630 CHAR1,15,16,BK$:GOTO1560
2640 :
2650 REM COPIA BLOCCO CARATTERI
2660 CHAR1,15,16,B$+"RETURN SCEGLIE, SPAZIO ESCE"
2670 CHAR1,16,22,LA$+"SCEGLI IL PRIMO CARATTERE "+ES$+"O"
2680 GOSUB1750:IFA$=" "THEN2630
2690 A=57344+(X+Y*32)*8+2048*SE
2700 CHAR1,21,22,LA$+"SCEGLI L' ULTIMO"+ES$+"O":GOSUB1750:IFA$=" "THEN2630
2710 B=57344+(X+Y*32)*8+2048*SE
2720 CHAR1,17,22,LA$+"COPIA SULLO STESSO SET ?"+ES$+"O"
2730 GETKEYA$:IFA$<>"S"ANDA$<>"N"THEN2730
2740 IFA$="N"THENBEGIN
2750 SE=XOR(SE,1):CHAR1,22,18,B$+"SET "+MID$("MAIMIN",1+3*SE,3)+"USCOLO"
2760 GOSUB2980:BEND
2770 CHAR1,15,22,LA$+"SCEGLI LA POSIZIONE DI COPIA"+ES$+"O"
2780 GOSUB1750:IFA$=" "THEN2630
2790 C=57344+(X+Y*32)*8+2048*SE
2800 CHAR1,15,22,BK$:IFB<AOR(B-A)>61440-C+2048*(SE=0)OR(C>AANDC<B)THEN2630
2810 CHAR1,21,22,LA$+"STO COPIANDO ..."+ES$+"O"
2820 BANK0:FORI=ATOB+7:POKEC+I-A,PEEK(I):NEXT:BANK15:CHAR1,15,22,BK$
2830 IFKI=1THENSYS3633
2840 GOTO2660
2850 :
2860 REM INVERTE BLOCCO CARATTERI
2870 CHAR1,15,16,B$+"RETURN SCEGLIE, SPAZIO ESCE"
2880 CHAR1,16,22,LA$+"SCEGLI IL PRIMO CARATTERE "+ES$+"O"
2890 GOSUB1750:IFA$=" "THEN2630
2900 A=57344+(X+Y*32)*8+2048*SE
2910 CHAR1,21,22,LA$+"SCEGLI L' ULTIMO"+ES$+"O":GOSUB1750:IFA$=" "THEN2630
2920 B=57344+(X+Y*32)*8+2048*SE
2930 CHAR1,15,22,BK$:IFB<ATHEN2630
2940 CHAR1,21,22,LA$+"STO LAVORANDO ..."+ES$+"O"
2950 BANK0:FORI=ATOB+7:POKEI,255-PEEK(I):NEXT:BANK15
2960 CHAR1,15,22,BK$:IFKI=1THENSYS3633
2970 GOTO2870
2980 CHAR1,45,16,CHR$(142-128*SE)+RE$
2990 FORI=0TO1:FORII=0TO3:PRINTTAB(45)CHR$(146-128*I);:FORCH=CH(II)TOCH(II)+31
3000 PRINTCHR$(CH);:NEXT:PRINT:NEXT:NEXT:PRINTCHR$(142);:RETURN
3010 CHAR1,5,16,CHR$(142)+C$:FORI=0TO7:PRINTTAB(5);
3020 FORCH=0TO7:PRINT".";:NEXT:PRINT:NEXT:RETURN
3030 :
3040 REM HPOKE PO,BY
3050 SYSSY,POANDDD,LO:SYSSY,INT(PO/DT),HI:SYSSK,BY:RETURN
3060 :
3070 REM R=HPEEK(PO)
3080 SYSSY,INT(PO/DT),HI:SYSSY,POANDDD,LO:SYSRE:R=PEEK(SI):RETURN
3090 :
3100 REM ERROR TRAPPING
3110 PRINTES$HO$HO$CH$:PRINT"ERRORE : "ERR$(ER):PRINT"STATO DISCO : "DS$
3120 FORI=0TO3000:NEXT:RESTORE1510:RESUME 1360
```

Simulatore del videogame "Frogger"

Un breve gioco, basato sulla tecnica dell'interrupt, che ricorda le imprese del povero ranocchio.

di Maurizio Dell'Abate

Per chi vuol solo giocare...

Copiate attentamente il listato, prestando particolare attenzione alle ultime righe di DATA, quindi salvate il programma (a scopo cautelativo) su disco o nastro PRIMA di dare il Run.

Lo scopo del gioco è molto semplice: attraversare una strada percorsa da ben otto auto da Formula 1 che sfrecciano in tutte le direzioni a diverse velocità; inutile dire che è necessario evitarle, al fine di non venire arrotati con conseguente perdita di una delle cinque vite a disposizione.

Per muovere l'omino (stavolta non è un ranocchio...) si utilizza il joystick in porta 1 oppure la tastiera: premendo il pulsante di fuoco si avanza di un passo, mentre muovendo a sinistra o a destra si provoca lo spostamento nelle corrispondenti direzioni. Si noti che, come ogni buon pedone dovrebbe sapere, non è possibile tornare indietro una volta entrati nella mischia dei bolidi.

Se non possedete un joystick potete utilizzare i tasti corrispondenti, ovvero CTRL e tasto "2" per spostamenti laterali e la barra (spazio), che simula il Fire Button.

Nell'angolo in alto a destra dello schermo si trovano tre indicatori di cui descriviamo le funzioni:

V - VELOCITA': indica la velocità media delle F1 che, col passare del tempo, aumenta costantemente e rapidamente; all'inizio della partita la velocità è uguale a 100 km/h (leggi: lumache), ma aumenta fino ad un massimo di 452 km/h (leggi: se riesci a passare sarai famoso). Ovviamente non si deve perdere tempo e il perchè dovreste capirlo da soli...

O - OMINI: è il numero delle vite di riserva, (5 alla partenza); si può perdere la vita in due modi: investiti da una F1 oppure cercando di recarsi oltre i limiti laterali dello schermo.

P - PUNTEGGIO: i punti aumentano di tre unità ad ogni passo in avanti; con l'attraversamento della strada si ottengono ben 50 punti di bonus.

...e per chi vuole imparare a programmare

Il gioco, come già accennato, è imperniato sulla programmazione dell'interrupt. Sui fascicoli arretrati (e futuri) di C.C.C. troverete dettagli su questo argomento, di cui diamo comunque una semplicissima descrizione.

L'interrupt

L'interrupt (interruzione) è un segnale elettronico che obbliga il microprocessore a sospendere tutto ciò che sta facendo e ad eseguire un programma in linguaggio macchina: al termine di questo programma è presente l'istruzione RTI (ReTurn from Interrupt) che riporta il micro a continuare il lavoro che aveva interrotto, come se niente fosse accaduto.

Le fonti che possono generare un'interruzione sono molteplici; la più importante è quella del TIMER A del CIA (uno dei due chip di I/O del C-64/128). Il CIA interrompe il micro ogni 60mo di secondo e lo obbliga a saltare all'indirizzo puntato dal vettore contenuto nelle locazioni 788 - 789 ($0314 - $0315) che, all'accensione del calcolatore, corrisponde a 59953 ($EA31). Qui risiedono le routine d'interrupt del Kernal (il sistema operativo) che assolvono funzioni indispensabili per il corretto funzionamento del computer tra cui la scansione della tastiera, l'incremento degli orologi, il lampeggio del cursore ed altre.

Modificando opportunamente il vettore d'interrupt, cioè settandolo per puntare ad una routine in RAM scritta dall'utente, si possono creare interessanti effetti di esecuzione pseudo-parallela, come ad esempio la generazione di una musica mentre stiamo programmando.

La nostra routine dovrà terminare con un JMP $EA31, in modo che venga eseguita anche la normale procedura d'interruzione.

Nel caso del gioco proposto, abbiamo realizzato una routine in L.M. assai breve che ha il compito di incrementare o decrementare la coordinata X degli otto sprite (le Formula 1). Osservando attentamente il disassemblato, dovreste capire tutto quanto sopra esposto ed il metodo che si è utilizzato per differenziare la velocità delle macchine.

Un'ultima nota: la locazione del CIA 56325 ($DC05) contiene la frequenza di invio dell'interruzione (in realtà non è proprio così, ma fa lo stesso): più il suo contenuto è basso, maggiore sarà la frequenza. Intervenendo su

56325 si è potuta regolare la velocità globale delle autovetture. Provate a digitare in modo diretto:

POKE56325,20

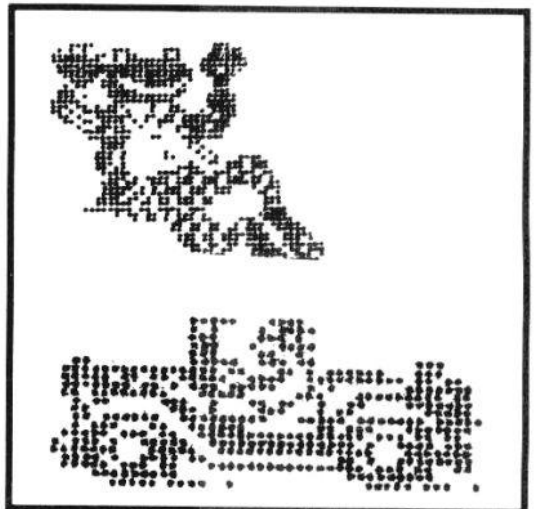
Il cursore lampeggerà più velocemente; la causa non ve la diciamo perchè ormai siete, o almeno dovreste essere, in grado di scoprirla da soli.

Il disassemblato

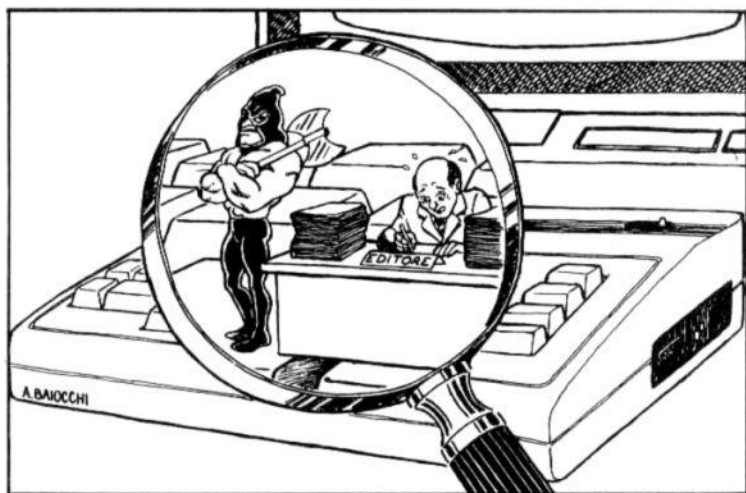
Ecco, di seguito, il disassemblato della routine in linguaggio macchina posta nell'interrupt. Il contenuto della locazione $A2 (162) viene incrementato dal Kernal ogni volta che si verifica un'interruzione. La routine effettua un AND fra il contenuto di $A2 ed 1 (uno); quindi, per mezzo di un BNE, fa in modo che le coordinate di due sprite vengano modificate OGNI DUE interruzioni (dando luogo, di conseguenza, ad uno spostamento più lento).

```
02A7 CE 00 D0 DEC $D000  muove sprite 0 a sinistra di un dot
02AA EE 02 D0 INC $D002  muove sprite 1 a destra di un dot...
02AD EE 02 D0 INC $D002  ...e di un altro dot = piu' veloce
02B0 EE 04 D0 INC $D004  muove sprite 2 a destra di un dot
02B3 CE 08 D0 DEC $D008  muove sprite 4 a sinistra di un dot
02B6 EE 0C D0 INC $D00C  muove sprite 6 a destra di un dot
02B9 CE 0E D0 DEC $D00E  muove sprite 7 a sinistra di un dot...
02BC CE 0E D0 DEC $D00E  ...e di un altro dot (piu' veloce)
02BF A5 A2    LDA $A2    accumulatore = PEEK($A2)
02C1 29 01    AND #$01   accumulatore = accumulatore AND 1
02C3 D0 06    BNE $02CB  se accum. <> 0 continua interrupt
02C5 EE 06 D0 INC $D006  muove sprite 3 a destra di un dot
02C8 CE 0A D0 DEC $D00A  muove sprite 5 a sinistra di un dot
02CB 4C 31 EA JMP $EA31  continua il programma d'interrupt 1
```

```
100 REM  MINI - FROGGER
110 REM  SOLO PER C/64/128
120 :
130 REM  BY M. DELL'ABATE
140 :
150 X1$=CHR$(17)+CHR$(157)+CHR$
    (157)+CHR$(157):X2$=X1$+CHR
    $(157)+CHR$(157)
170 SYS65409:REM  RESET DEL VID
    EO
180 FOR A=832 TO 832+62:READ B:
    POKE A,B:NEXT
190 FOR A=896 TO 896+62:READ B:
    POKE A,B:NEXT
200 FOR A=679 TO 717:READ B:POK
    E A,B:NEXT
210 V=53248
220 FOR A=0 TO 7:NC=RND(1)
230 IF NC<.5 THEN POKE V+39+A,1
    :GOTO 260
240 IF NC<.7 THEN POKE V+39+A,7
    :GOTO 260
250 POKE V+39+A,0
260 NEXT
270 PN=2040:POKE PN,14:POKE PN+
    1,13:POKE PN+2,13:POKE PN+3
    ,13:POKE PN+4,14
280 POKE PN+5,14:POKE PN+6,13:P
    OKE PN+7,14
290 REM  ↑↑ PUNTATORI E COLORI
    SPRITES
300 S=54272:FOR A=0 TO 24:POKE
    S+A,0:NEXT
310 POKE S+24,15:POKE S+6,240
320 POKE S,35:POKE S+1,25
330 GOTO 350
340 POKE S+4,17:FOR TE=0 TO 20:
    NEXT:POKE S+4,16:RETURN
350 GOSUB 340:POKE 53281,12:POK
    E 53280,15:PRINT:POKE 646,7
360 PRINT" IL PEDONE..."
370 PRINT:PRINT:POKE 646,0:PRIN
    T" GAME PER C-64 BY M. DELL
    'ABATE"
380 POKE 646,1:PRINT:PRINT:PRIN
    T:PRINT:PRINT:PRINT" PREMI
    UN TASTO PER GIOCARE"
390 POKE 198,0:WAIT 198,1:POKE
    198,0:GOSUB 340
400 PRINTCHR$(147):IC=70
410 FOR A=0 TO 14 STEP 2
420 POKE V+A,256*RND(1)
```

```
430 POKE V+A+1,IC:IC=IC+15:NEXT
440 POKE 56334,PEEK(56334) AND
    254
450 POKE 788,167:POKE 789,2
460 POKE 56334,PEEK(56334) OR 1
470 POKE V+21,255:POKE 646,0
480 VI=5:PO=90:PT=0
490 POKE 1096,22:POKE 1136,15:P
    OKE 1176,16
500 X=14:Y=21
510 CO=PEEK(V+31):GOTO 610
520 POKE 56325,PO
530 POKE 211,33:POKE 214,1:SYS5
    8640:PRINTINT(9000/PO);X2$;
    VI;X1$;PT
540 SS=GG:EF=PEEK(56321)
550 IF PEEK(V+31) THEN 690
560 IF PO>20 THEN PO=PO-.1
570 IF EF=239 THEN Y=Y-1:PT=PT+
    3:GOTO 610
580 IF EF=251 THEN X=X-1:GOTO 6
    10
590 IF EF=247 THEN X=X+1:GOTO 6
    10
600 GOTO 520
610 GG=1024+X+40*Y
620 POKE GG,81:POKE S+GG,1
630 POKE SS,32
640 IF X<1 OR X>28 THEN 690
650 IF Y=1 THEN 670
660 GOTO 540
670 FOR A=0 TO 9:GOSUB 340:NEXT
680 PT=PT+50:POKE GG,32:GOTO 50
    0
690 POKE 56334,PEEK(56334) AND
    254
700 FOR A=1 TO 10:GOSUB 340
710 FOR B=2 TO 3:POKE 53265,(PE
    EK(53265) AND 248) OR B:NEX
    T:NEXT:POKE GG,32
720 POKE 56334,PEEK(56334) OR 1
730 VI=VI-1:IF VI=0 THEN PRINT"
    [BIANCO][HOME][DOWN] GAME O
    VER":POKE 1138,48:FOR G=0 T
    O 3000:NEXT: RUN
740 GOTO 500
750 :
760 :
770 DATA  0,0,0,0,0,0,0,0,0,0,0
    ,0,3,192,0,3,192
780 DATA  56,27,223,56,27,223,1
    84,25,183,240,31,238,252,27
    ,239,63,31
790 DATA  238,252,25,183,240,27
    ,223,184,27,223,56,3,192,56
    ,3,192,0
800 DATA  0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0
810 DATA  0,0,0,0,0,0,0,0,3,192
    ,28,3,192,28,251,216,29
820 DATA  251,216,15,237,152,63
    ,119,248,252,247,216,63,119
    ,248,15,237,152
830 DATA  29,251,216,28,251,216
    ,28,3,192,0,3,192,0,0,0,0,0
840 DATA  0,0,0,0,0,0,0
850 :
860 :
870 DATA 206,0,208,238,2,208,23
    8
880 DATA 2,208,238,4,208,206,8
890 DATA 208,238,12,208,206,14,
    208
900 DATA 206,14,208,165,162,41,
    1
910 DATA 208,6,238,6,208,206,10
920 DATA 208,76,49,234
930 :
940 REM  DI M. DELL'ABATE (C)
```

Un'occhiata all'esecutore ed una all'editore

Studiando attentamente l'interprete Basic vengon fuori autentiche sorprese...

di Claudio Baiocchi

Le parole riservate del Basic si distinguono in due categorie:
• quelle che possono essere poste all'inizio di una linea Basic, o subito dopo i "due punti" (come Let, Data, If, ed altre).
• quelle che, se poste in tale posizione, provocano un "SYNTAX ERROR" (TAN, LEFT$, AND, STEP).

Alle parole della prima categoria si suole dare il nome di "Comandi".

In fase di editing, cioè quando digitiamo una linea di programma o una frase da eseguire immediatamente, ogni parola chiave viene TOKENIZZATA, cioè sostituita col suo numero di "codice" (detto appunto TOKEN); la tabella del riquadro 1 riporta per ogni parola chiave il token corrispondente.

Prime difficoltà

Come si può notare dalla tabella stessa, il Basic 2.0 (quello del Vic 20 e del C/64), pur essendo più povero, è molto più "ordinato" delle versioni successive: in effetti, indipendentemente da ciò che avviene in fase di listing, nel Basic 2.0 la tabella contiene prima tutti i comandi (token da 128 a 162), poi le altre parole chiave (token da 163 a 202).

Uniche eccezioni sono il token 203 (codice di GO) ed il 255, token di pi greco.

Nel Basic 3.5 (quello di C/16 e Plus/4) e nel Basic 7.0, (quello del C/128), i comandi vanno invece da 128 a 162, poi da 213 a 250. Il 202 (token di MID$) può anche essere un comando (?); ciò si ripercuote naturalmente in una minore velocità di elaborazione, ma d'altronde l'alternativa era distruggere ogni forma di compatibilità.

Come ragiona l'interprete

Il Basic Commodore ha due tipi di "separatori": il 58 (token dei "due punti", che separa più istruzioni poste sulla stessa linea) e lo 0 (token di fine linea). Non c'è pericolo di confusione con la cifra zero (0), tokenizzata con 48, suo codice ASCII.

Seguiamo passo passo il lavoro che l'interprete svolge per eseguire il singolo statement, cioè un blocco di istruzioni compreso tra due separatori (la routine in questione è spesso detta NSE, Next Statement Executor). Ci riferiamo, per ora, al Vic 20 e al C/64: torneremo più in là su qualche variante relativa ai modelli successivi.

Si tenga presente che i singoli token da eseguire vengono letti tramite ciò che in Basic potrebbe essere descritto da:

```
I=PEEK(122)+256*PEEK(123):
T=PEEK(I)
```

(I è l'Indirizzo della cella contenente il token da eseguire; T è il valore di tale Token); supponendo che T sia il token di un separatore (è a tale punto che va spezzato il ciclo per una sua corretta descrizione) le fasi della routine NSE si svolgono nel seguente modo :

(I) se PEEK(123)>2 si esegue un'operazione che in Basic potrebbe essere descritta con:

```
POKE61,PEEK(122):
POKE62,PEEK(123)
```

Token e parole chiave corrispondenti

Su tutti i modelli, tra i codici inferiori a 128, solo lo 0 (= fine linea) e il 58 (=doppio punto) rappresentano delle istruzioni; ci limitiamo perciò ai token da 128 a 255.

Il simbolo "C" significa "Comando". La colonna "Versioni Basic" indica la versione in cui è presente la parola stessa: V2.0 (Vic 20 e C/64); 3.5 (C/16 e Plus/4) e 7 (C/128). La colonna "Solo List" fa riferimento al Basic 2.0 (Vic 20 e Plus/4) e fornisce l'effetto del token in questione in sede di LIST; in sede di esecuzione tale token genera errore.

#Parola	Versione Basic
128 END	C Tutte
129 FOR	C Tutte
130 NEXT	C Tutte
131 DATA	C Tutte
132 INPUT•	C Tutte
133 INPUT	C Tutte
134 DIM	C Tutte
135 READ	C Tutte
136 LET	C Tutte
137 GOTO	C Tutte
138 RUN	C Tutte
139 IF	C Tutte
140 RESTORE	C Tutte
141 GOSUB	C Tutte
142 RETURN	C Tutte
143 REM	C Tutte
144 STOP	C Tutte
145 ON	C Tutte
146 WAIT	C Tutte
147 LOAD	C Tutte
148 SAVE	C Tutte
149 VERIFY	C Tutte
150 DEF	C Tutte
151 POKE	C Tutte
152 PRINT•	C Tutte
153 PRINT	C Tutte
154 CONT	C Tutte
155 LIST	C Tutte
156 CLR	C Tutte
157 CMD	C Tutte
158 SYS	C Tutte
159 OPEN	C Tutte
160 CLOSE	C Tutte
161 GET	C Tutte
162 NEW	C Tutte
163 TAB(	Tutte
164 TO	Tutte
165 FN	Tutte
166 SPC(	Tutte
167 THEN	Tutte
168 NOT	Tutte
169 STEP	Tutte
170 +	Tutte
171 -	Tutte
172 *	Tutte
173 /	Tutte
174 •	Tutte
175 AND	Tutte
176 OR	Tutte
177 >	Tutte
178 =	Tutte
179 <	Tutte
180 SGN	Tutte
181 INT	Tutte
182 ABS	Tutte
183 USR	Tutte
184 FRE	Tutte
185 POS	Tutte
186 SQR	Tutte
187 RND	Tutte
188 LOG	Tutte
189 EXP	Tutte
190 COS	Tutte
191 SIN	Tutte
192 TAN	Tutte
193 ATN	Tutte
194 PEEK	Tutte
195 LEN	Tutte
196 STR$	Tutte
197 VAL	Tutte
198 ASC	Tutte
199 CHR$	Tutte
200 LEFT$	Tutte
201 RIGHT$	Tutte

#Parola	Versione Basic	Solo LIST	Note
202 MID$	Tutte		(1)
203 GO	C Tutte		(2)
204 RGR	3.5/7		(3)
205 RCLR	3.5/7	FOR	
206 RLUM	3.5	NEXT	(4)
207 JOY	3.5/7	DATA	
208 RDOT	3.5/7	INPUT•	
209 DEC	3.5/7	INPUT	
210 HEX$	3.5/7	DIM	
211 ERR$	3.5/7	READ	
212 INSTR	3.5/7	LET	
213 ELSE	C 3.5/7	GOTO	
214 RESUME	C 3.5/7	RUN	
215 TRAP	C 3.5/7	IF	
216 TRON	C 3.5/7	RESTORE	
217 TROFF	C 3.5/7	GOSUB	
218 SOUND	C 3.5/7	RETURN	
219 VOL	C 3.5/7	REM	
220 AUTO	C 3.5/7	STOP	
221 PUDEF	C 3.5/7	ON	
222 GRAPHIC	C 3.5/7	WAIT	
223 PAINT	C 3.5/7	LOAD	
224 CHAR	C 3.5/7	SAVE	
225 BOX	C 3.5/7	VERIFY	
226 CIRCLE	C 3.5/7	DEF	
227 GSHAPE	C 3.5/7	POKE	
228 SSHAPE	C 3.5/7	PRINT•	
229 DRAW	C 3.5/7	PRINT	
230 LOCATE	C 3.5/7	CONT	
231 COLOR	C 3.5/7	LIST	
232 SCNCLR	C 3.5/7	CLR	
233 SCALE	C 3.5/7	CMD	
234 HELP	C 3.5/7	SYS	
235 DO	C 3.5/7	OPEN	
236 LOOP	C 3.5/7	CLOSE	
237 EXIT	C 3.5/7	GET	
238 DIRECTORY	C 3.5/7	NEW	
239 DSAVE	C 3.5/7	TAB(	
240 DLOAD	C 3.5/7	TO	
241 HEADER	C 3.5/7	FN	
242 SCRATCH	C 3.5/7	SPC(	
243 COLLECT	C 3.5/7	THEN	
244 COPY	C 3.5/7	NOT	
245 RENAME	C 3.5/7	STEP	
246 BACKUP	C 3.5/7	+	
247 DELETE	C 3.5/7	-	
248 RENUMBER	C 3.5/7	*	
249 KEY	C 3.5/7	/	
250 MONITOR	C 3.5/7	•	
251 USING	3.5/7	AND	
252 UNTIL	3.5/7	OR	
253 WHILE	3.5/7	>	
254	nessuna	=	(5)
255 PI GRECO	Tutte		

NOTE

(1) su C/16, Plus/4 e C/128 la parola MID$ può anche essere un comando; la relativa routine è gestita in modo anomalo.

(2) Su tutti i modelli il comando GO è gestito in modo anomalo (sul C/128, oltre che da TO, esso può anche essere seguito da 64).

(3) La funzione RGR non è implementata nel Basic 2.0; l'inserimento di tale codice nel programma genera però effetti strani; si veda l'articolo.

(4) Checchè ne dica il manuale (pagina K-4 della edizione in inglese) la funzione RLUM non è implementata sul C/128: il valore 206 è usato per segnalare che il successivo codice va gestito secondo la tabella aggiuntiva seguente:

206	1	POT
206	2	BUMP
206	3	PEN
206	4	RSPPOS
206	5	RSPRITE
206	6	RSPCOLOR
206	7	XOR
206	8	RWINDOW
206	9	POINTER

(5) Su C/16 e Plus/4 il codice 254 dà errore; sul C/128 il 254 è usato per segnalare che il successivo codice va gestito secondo la tabella aggiuntiva seguente (sono tutti dei comandi) :

254	2	BANK
254	3	FILTER
254	4	PLAY
254	5	TEMPO
254	6	MOVSPR
254	7	SPRITE
254	8	SPRCOLOR
254	9	RREG (6)
254	10	ENVELOPE
254	11	SLEEP
254	12	CATALOG
254	13	DOPEN
254	14	APPEND
254	15	DCLOSE
254	16	BSAVE
254	17	BLOAD
254	18	RECORD
254	19	CONCAT
254	20	DVERIFY
254	21	DCLEAR
254	22	SPRSAV
254	23	COLLISION
254	24	BEGIN(7)
254	25	BEND
254	26	WINDOW
254	27	BOOT
254	28	WIDTH
254	29	SPRDEF
254	30	QUIT (8)
254	31	STASH
254	33	FETCH
254	35	SWAP
254	36	OFF (8)
254	37	FAST
254	38	SLOW

(6) Il comando RREG, citato a pagina 20-3 ed a pagina K-7, non è documentato nell'Encyclopaedia; la sintassi prevede che RREG sia seguita dal nome di una, due, tre o quattro variabili, nelle quali vengono immessi nell'ordine i valori dei registri interni P,A,X,Y.

(7) BEGIN, che avrebbe tutto il diritto di essere un comando, non è accettato se non dopo l'apertura di una IF

(8) QUIT e OFF non sono implementati; ma sono accettati in fase di editing, e regolarmente listati. In fase di esecuzione generano il messaggio di non implementazione, e non un generico "SYNTAX ERROR".

Sempre in contrasto con i manuali, si ha che GET# e GETKEY non sono veri e propri comandi: la loro gestione avviene come sottocaso della gestione di GET; analogamente sul C/128 il comando GO64 non ha un token proprio, e viene gestito nell'ambito di GO; infine le parole COLINT, MOVESHAPE, RSPR citate nel manuale del C/128, Appendice K, non esistono; le altre parole riservate (TI, TI$, ST, DS, DS$, ER, EL) non hanno un token proprio, sono codificate in forma ASCII standard, e vengono gestite a parte.

(II) se T=58 si salta alla fase (V); se T=0 si va alla fase (III); negli altri casi si chiama la routine di errore: il lavoro deve cominciare a partire da un separatore!

(III) giunti qui, è PEEK(I)=0; se risulta PEEK(I+2)=0 si salta alla routine del READY: il programma è finito.

(IV) si esegue l'equivalente dell'operazione Basic:
POKE57,PEEK(I+1):
POKE 58,PEEK(I+2)
poi si pone I=I+3 (ovviamente lavorando sulle memorie 122 e 123: le variabili I e T esistono solo come abbreviazione di linguaggio)

(V) si guarda se è stato premuto il tasto STOP; in caso affermativo si esegue la routine relativa.

(VI) se è T=0 oppure T=58 si torna alla fase (I); altrimenti si va in subroutine alla fase (VII) e, al rientro dalla subroutine, si torna alla fase (I)

(VII) se risulta T<128 si assume che sia stato soppresso il comando opzionale Let, e si salta alla routine relativa (sarà tale routine che verificherà la correttezza della assunzione)

(VIII) se è T<=162 si esegue l'analogo del Basic:
ON T-128 GOTO ..... (35 indirizzi)

(IX) se è T=203 (token di GO) e se è PEEK(I+1)=164 (token di TO) si salta alla routine del GOTO

(XI) si salta alla routine di errore: c'è uno statement che non inizia con un comando.

Naturalmente, nelle fasi (VII), (VIII), (IX) la singola routine chiamata incrementerà "I" per leggere i parametri necessari. Se non ci sono errori la routine stessa terminerà con un RTS (analogo in L.M. del comando RETURN) restituendo la mano con un valore di "I" corrispondente ad un separatore, cosicchè il ciclo può riiniziare.

Nella gestione di tali routine si farà ancora qualcosa di analogo per quanto riguarda i codici da 180 a 202, che corrispondono alle cosiddette FUNZIONI; non ci occuperemo della gestione dei codici tra 163 e 179 per non appesantire troppo la trattazione...

Il codice stesso viene raddoppiato, ed in funzione del valore ottenuto si legge in una apposita tabella l'indirizzo della routine cui saltare per l'esecuzione della funzione voluta.

La tabella in ROM di cui parliamo, contiene, nell'ordine, la parte bassa e la parte alta dell'indirizzo; ad ogni codice corrispondono perciò due elementi della tabella, e questo è il motivo per cui il codice viene inizialmente raddoppiato. Guardando in dettaglio la routine che gestisce lo smistamento delle funzioni si può tuttavia notare che essa contiene un BUG (=errore): se il token incontrato è maggiore di 199 (i "veri" codici in questione corrispondono a LEFT$, RIGHT$, MID$, cioè alle funzioni con più di un parametro) NON viene effettuato il controllo che il codice sia minore di 203. In casi come questo l'interprete salta direttamente all'indirizzo fornito dalla tabella, con risultati talvolta disastrosi dal momento che la tabella stessa, non è più affidabile per valori del token superiori a 202.

Le prime sorprese

Per fortuna il bug non è pericoloso (a meno che non si siano fatte poke "strane" nel programma) e token da 204 in poi non dovrebbero essere incontrati. Vi sono, però, due eccezioni:
• la prima è il tentativo di eseguire sul C/64 programmi scritti per il C/16, il Plus/4 oppure il C/128;
• la seconda è collegata al codice 203, token del GO, che può essere stato inserito per errore in un programma.

Eseguendo una frase del tipo:
PRINT GO(A$,2)
il computer, anzichè emettere un Syntax Error, esegue un salto alla locazione RAM 27001, con risultati imprevedibili!

Un problema analogo avviene in fase di LIST: nel Basic 2.0 la "de-tokenizzazione" (=traduzione) avviene inviando alla routine di stampa, uno dopo l'altro, i caratteri prelevati da una certa tabella.

All'interno di questa, la fine di ogni singola parola chiave è segnalata da una lettera scritta in reverse (che sarà comunque stampata normale).

Per risparmiare un byte di programma nella routine che gestisce il LIST, i progettisti della Commodore hanno strutturato la routine tramite un ciclo che, dopo la stampa del singolo carattere, termina con l'istruzione BNE (cioè: se il carattere stampato non è uno 0, continua il tuo lavoro, altrimenti prosegui). Naturalmente tutti i codici inviati alla stampa in condizioni normali sono diversi da zero; e ciò resta vero anche per i codici da 205 in poi. Al contrario, nella de-tokenizzazione del 204, il primo carattere inviato alla stampa è in realtà lo 0 di fine tabella; così, dopo aver eseguito la stampa di un (illeggibile) CHR$(0), la routine di LIST, invece di tornare indietro ad eseguire il suo lavoro, va avanti inserendosi nella routine successiva (che è quella che gestisce il comando FOR); col risultato di generare un Syntax Error. Un'indicazione di come sfruttare tale bug per proteggere i programmi è indicato nel riquadro 2.

Token diversi dal 204 non generano errore, ma provocano listati inattendibili: l'elenco di come (in fase di listing) il Basic 2.0 interpreta i token dei Basic 3.5 e 7.0 è fornito nella tabella del riquadro 1, alla colonna "Solo List".

Altri casi imbarazzanti

Torniamo ora alla distinzione di casi (VII, VIII, IX, X, XI) svolta dall'esecutore: essa dovrebbe essere chiaramente motivata da quanto detto relativamente ai token dei comandi nel Basic 2.0; così come dovrebbe essere chiaro che la casistica relativa ai Basic più evoluti sarà più complicata; e d'altronde la complicazione è spesso fonte di errori.

Esplicitiamo un bug, per fortuna non grave, presente su C/16, Plus/4 e C/128, legato alla gestione delle funzioni (token compresi tra 180 e 212): eseguito il controllo che il numero in

questione sia tra 180 e 212, e raddoppiato ancora il token da trattare, si fa ricorso alla solita tabella di smistamento; ma bisogna eliminare il caso corrispondente al GO...; e qui i progettisti Commodore sono stati attratti dal fascino del complicato: invece di porre nella tabella, in corrispondenza all'entrata N.203, l'indirizzo della routine di errore, hanno pensato bene di scalare di un posto (due byte) tutti gli indirizzi in tabella corrispondenti a codici superiori a 203; e di gestire i codici superiori a 203 tramite un iniziale decremento del codice stesso!

In realtà, oltre ad uno spreco di tempo e di spazio (due byte in più nella tabella "costano meno" del confronto e della corrispondente decisione) si è anche ottenuto di trattare il codice 203 come se fosse un 202; col risultato che su C/16, Plus/4 e C/128 la frase:

A$=GO(B$,X,Y)

non genera un Syntax Error (nè un rinvio a routine RAM come su Vic 20 e C/64), ma viene tranquillamente eseguita come se fosse stata scritta:

A$=MID$(B$,X,Y)

Come ottenere listati più belli o... non listabili

Tutti sanno che lo scopo del comando REM è quello di permettere di inserire commenti in un programma, senza disturbare l'esecuzione del programma stesso; non tutti sanno invece che, sempre senza pregiudicare l'esecuzione del programma, si possono inserire commenti anche in altri modi: ad esempio il seguente programma gira senza problemi sul C/64 (sul Vic 20 basterà sostituire il 43256 di linea 100 con 51448; sul C/128 con 21135; su C/16 e Plus/4 con 36272):

```
10 GOSUB 100 E NON SCRIVERE SYNTAX ERROR :PRINT "O.K."

50 GOTO 200 SENZA PROTESTARE
100 SYS 43256 QUI SI PUO' SCRIVERE CIO' CHE SI VUOLE:RETURN
200 PRINT "SEMPRE TUTTO O.K."
```

La misteriosa SYS di linea 100 chiama la routine ROM che gestisce il comando DATA che si limita, semplicemente, a cercare, dal punto in cui è stata chiamata, la prima cella di memoria che contiene un 58 (token dei "due punti") o uno 0 (token di fine linea).

E' a partire da tale cella che l'esecutore riprenderà il suo lavoro. Il perchè la riga 50 non causa problemi dovrebbe essere ovvio. Per quanto concerne infine la linea 10 si tenga presente che la routine del RETURN, dopo aver calcolato l'indirizzo di ritorno, cede la mano alla routine del DATA, che provvede appunto ad incrementare tale indirizzo fino a trovare uno 0 oppure un 58.

Possiamo approfittare di quanto visto per migliorare l'estetica dei nostri listati. Se, ad esempio, volete evidenziare una linea importante del programma, potete forzare il computer a listarla in Reverse, procedendo come segue. Sia da evidenziare la linea:

```
500 FOR X=1 TO 100:REM CICLO PRINCIPALE
```

Si scriva tale linea nella forma:

```
500 SYS 43256 %% : FOR X=1 TO 100:REM CICLO PRINCIPALE
```

e si aggiungano provvisoriamente le linee :

```
498 FOR X = PEEK(61)+256*PEEK(62) TO X + 1000 : IF PEEK(X)*PEEK(X+1) <> 37#2 THEN NEXT
499 POKE X,13 : POKE X+1,18 : END
```

Eseguito:RUN 498 (return)

basterà poi sopprimere le linee 498, 499 ed il gioco è fatto: la POKE ha sostituito i due token 37 (corrispondenti ai due simboli di percentuale %%) con 13 (token dell'"a capo") e con 18 (token della scrittura in reverse.

L'uso delle locazioni 61, 62 è motivato nell'articolo; il trucco, così come è descritto, funziona sul C/64, ma le modifiche da apportare per gli altri modelli sono semplici: il diverso indirizzo della SYS è già stato indicato; su C/16, Plus/4 e C/128 non si può fare uso delle locazioni 61, 62 (gestite in modo diverso) ma basta far partire il ciclo da inizio Basic.

Su Vic 20 e C/64 possiamo anche adattare quanto visto alla protezione di programmi: un trucco già segnalato su questa rivista è quello di inserire come prima linea del programma:

0 REM L

dove il carattere che segue la REM (che sembra una "L" maiuscola) si ottiene invece premendo il tasto Shift e contemporaneamente la lettera L. In effetti tale carattere, corrispondente al token 204, provoca un SYNTAX ERROR in sede di listing: al comando LIST seguirà solo :

0 REM

seguito da "SYNTAX ERROR".

E' tuttavia facile sbarazzarsi di questa rudimentale protezione: basta eseguire

LIST 1-

e si legge tutto il resto del programma oppure, più semplicemente, si può sopprimere la linea 0 che, iniziando con REM, non dovrebbe contenere parti essenziali del programma.

Un raffinamento del trucco si ottiene (su C/64; le varianti per il Vic 20 sono ovvie) inserendo le linee :

```
0 SYS 43256*:GOSUB23456:X=A(11)
...
...
...
23456 SYS 43256*:DIMA(11):READ A,B,C
23458 SYS 43256*:seguita da altre forme di protezione (disabilitazione del LIST, dello STOP, del Run/Stop & Restore...)
23460 SYS 43256*:RETURN
...
...
```

poi, tramite delle POKE, si sostituiranno gli asterischi col codice proibito 204. Si osservi che, fuori da una REM, il codice 204 va inserito nel programma tramite una POKE, e non digitando SHIFT+L: infatti, nella costruzione di linee Basic, la routine di tokenizzazione sopprime i caratteri semi-grafici non preceduti da REM o non racchiusi tra virgolette.

E' chiaro che ora la soppressione pura e semplice delle linee che generano un "SYNTAX ERROR provocherà guai notevoli in sede di esecuzione...

con risultati non sempre identificabili.

Resta da capire il senso delle fasi (I), (III), (IV) del lavoro dell'esecutore. Per quanto concerne la fase (I) (che esiste solo nel Basic 2.0) si osservi che PEEK(123)=2 significa che si sta interpretando un pezzo di programma scritto nel cosiddetto Buffer bASIC; in altre parole si sta eseguendo un comando diretto, e non una linea di programma. Ne consegue che, su Vic 20 e C/64, durante l'elaborazione di un programma le memorie 61 e 62 "puntano" sempre all'ultimo separatore incontrato. Il Basic utilizza tale informazione in occasione, tra l'altro, del messaggio REDO FROM START per sapere da dove riprendere l'elaborazione; l'utente smaliziato può utilizzare la stessa informazione per vari scopi; un esempio è fornito nel riquadro 2.

Per quanto concerne i modelli successivi, il Basic 3.5 utilizza direttamente le memorie 61, 62 per leggere il token da interpretare (senza cioè fare uso delle memorie intermedie 122 e 123); sul C/128 la situazione è analoga, ma si usano le locazioni 63, 64 anzichè le 60, 62.

Per capire le fasi (III), (IV) del lavoro dell'esecutore abbiamo invece bisogno di qualche informazione sul lavoro dell'"editore". Cioè su che cosa avviene nella cosiddetta fase di "editing", quando digitiamo una linea di programma. Anche qui è bene spezzare il tutto in più fasi.

Quando si scrive un programma

Supponiamo di avere in memoria il programma (brevissimo, per semplificare):

```
1000 PRINT"CIAO"
1010 END
```

Per sapere come tale programma è memorizzato nella RAM del nostro Commodore, basta eseguire delle PEEK, a partire dalla cella precedente a quella "puntata" da 43,44 (su C/128 si sostituisca 43,44 con 45,46) cioè eseguire:

```
FOR X=PEEK(43)+256*PEEK(44)-1
TO X+50:
PRINT PEEK(X),:
NEXT (return)
```

Il numero 50 è scelto ad abundantiam: solo i primi dati saranno significativi. Interpretiamo i dati che verranno stampati.

(A) innanzitutto verrà mostrato uno 0; si tratta del codice di inizio Basic.

(B) Dopo tale 0 si troveranno due numeri che chiameremo provvisoriamente L1, L2; il loro valore dipende dal Commodore su cui si sta lavorando: sul C/64 si avrà 13,8; sul Vic 20 inespanso si avrà 13,16; sul C/128 si avrà 13,28; eccetera. Torneremo presto sul significato di questi due numeri, che si chiamano "link alla linea seguente".

(C) i due numeri successivi sono 232 e 3. Si tratta del numero di linea (1000 nel nostro caso) scritto nella solita forma parte bassa - parte alta: in effetti
232+256*3=1000.

(D) Solo adesso comincia il vero programma: il numero successivo è 153, codice del comando PRINT (si veda la tabella del riquadro 1); segue il numero 34, codice delle virgolette; poi i numeri 67, 73, 65, 79, codici di C, I, A, O rispettivamente; ancora un 34, codice delle virgolette; ed uno 0, codice di "fine linea".

(E) La situazione si ripete ora analoga a quella vista a partire da (B): si hanno due valori "di link", diciamo L3, L4; poi i numeri 242 e 3, rappresentazione in forma bassa - alta del numero di linea
(1010=242+256*3)
segue il numero 128, codice del comando END; e lo 0 di fine linea.

(F) Per analogia dovremmo ora aspettarci una coppia di numeri di link; però il programma è finito....; in effetti troviamo una coppia di zeri che, come vedremo, esprimono esattamente il fatto che il programma è finito.

Precisiamo ora il significato dei termini di link: qualunque sia il computer su cui stiamo lavorando, il numero L1+256*L2 fornisce la cella di inizio della prossima linea Basic (cioè la cella successiva a quella che contiene il codice zero di fine linea attuale); analogamente per L3+256*L4. In generale i due numeri di link sono perciò un puntatore (nella solita forma bassa-alta) che fornisce il numero d'ordine (in termine tecnico: l'indirizzo) della cella RAM dove inizia la linea Basic successiva. Tenendo conto del fatto che il programma Basic non può essere scritto in una zona RAM con indirizzo tra 0 e 255 (è la cosiddetta Pagina 0, riservata ad altri tipi di memorizzazioni) in condizioni normali il secondo dei valori di link non può essere uno 0; ed il Computer mette uno zero in tale posizione quando vuole segnalare che il programma è finito; risulta così chiarito il senso della fase (III) del lavoro dell'esecutore.

Dov'è la linea?

Chi conosce un po' di terminologia nell'ambito della rappresentazione dei dati non avrà difficoltà a riconoscere nella struttura che abbiamo appena descritto una "lista unidirezionale": ogni "dato" (=linea di programma) comincia con un puntatore al dato successivo. La lista è unidirezionale perchè non sono fornite indicazioni per risalire da una linea a quella precedente. Indipendentemente dalle proprie conoscenze sulle strutture dati, dovrebbe però essere ovvio un algoritmo per verificare se esiste una data linea e, in caso affermativo, costruirne l'indirizzo d'inizio: basterà sviluppare le fasi seguenti:

(1) si pone
I=PEEK(43)+256*PEEK(44)-1

sul C/128, invece:
I=PEEK(45)+256*PEEK(46)-1
(in quest'ultimo caso le PEEK che seguono dovranno essere fatte nel banco N. 0);

(2) se PEEK(I+2)=0 la lista è finita; si esce dal ciclo con risposta negativa

(3) se PEEK(I+3)+256*PEEK(I+4) è maggiore del numero cercato si esce dal ciclo con risposta negativa

(4) se PEEK(I+3)+256*PEEK(I+4) è minore del numero cercato, si pone I=PEEK(I+1)+256*PEEK(I+2)-1 e si torna alla fase (2).

(5) se si è arrivati qui, la "vera" linea cercata inizia nella cella I+5; da I+1 a I+4 si hanno informazioni complementari.

Si osservi che queste fasi vengono eseguite dal computer in varie situazioni: in seguito al comando LIST, nelle sue varie forme; in seguito a comandi quali RUN, GOTO, GOSUB e THEN seguiti da un numero di linea; in seguito a ON... GOTO e ad ON... GOSUB; e anche in fase di scrittura di nuove linee di programma.

In quest'ultimo caso il lavoro è improbo; occorrerà: trovare il giusto punto in cui inserire la nuova linea; eliminare un'eventuale linea già esistente con lo stesso numero, riaccostando poi le linee che seguono; fare spazio alla nuova linea spostando in avanti tutte quelle che seguono; inserire la linea (dopo la sua tokenizzazione) nello spazio creato. Vanno poi ricostruiti tutti i link; ed infine, salvo che sul C/128, viene eseguito un CLR. Per fortuna ci pensa il computer: noi non ce ne accorgiamo, salvo quando modifichiamo una delle prime linee di un programma molto lungo: la ricomparsa del cursore si fa allora aspettare, ed è giusto che sia così....

Quando si esegue un ordine

Occupiamoci ora di ciò che avviene in fase di esecuzione: una volta ottenuto il corretto valore I del punto di inizio dell'elaborazione (in seguito ad un "RUN" è semplicemente I=PEEK (43)+256*PEEK(44)-1; in seguto a GOTO, GOSUB o RUN seguiti da un numero di linea il valore di I è calcolato come visto prima) il comando delle operazioni viene preso dalla routine NSE descritta all'inizio dell'articolo; routine di cui siamo anzi ormai in grado di interpretare correttamente la fase (IV): all'inizio di ogni nuova riga Basic l'esecutore pone nelle memorie 57, 58 (sul C/128: nelle memorie 59 e 60) il numero della linea Basic che si appresta ad eseguire: in altri termini, durante l'esecuzione di un programma, le istruzioni:

NL=PEEK(57)+256*PEEK(58)

forniranno in NL il numero di linea attualmente in elaborazione mentre in fase di esecuzione di comandi diretti il contenuto della locazione 58 è 255, troppo grande per essere confuso con la parte alta di un numero di linea. L'uso che il computer fa di tali informazioni è, con un'unica eccezione, solo di tipo diagnostico: l'informazione è sfruttata per emettere i messaggi di errore corredati dal numero di linea in cui l'errore è stato incontrato. L'eccezione concerne la gestione di comandi del tipo "GOTO N", nella quale le fasi (1),..., (5) di ricerca della linea voluta elencate precedentemente vengono accelerate col seguente artificio: se risulta N<=PEEK(43)+256*PEEK(44) si procede esattamente come già visto; se, al contrario, risulta N>PEEK(43)+256*PEEK(44), la fase (1) viene modificata: per costruire il valore iniziale di I si procede in avanti, a partire dalla cella in esame, fino a trovare uno 0 (= fine linea); e si sceglie come I iniziale l'indirizzo della cella contenente tale 0.

Visto l'USO che il Computer fa delle memorie 57, 58, ci si può porre il problema di quali ABUSI può commettere su di esse l'utente smaliziato; alcuni di tali aspetti sono stati discussi nell'articolo "Due memorie molto strane", CCC N.29. Chi non ha letto tale articolo può dare un'occhiata al riquadro 3 qui in cui sono mostrati alcuni pericoli connessi alla manipolazione tramite POKE dei numeri di linea.

Per i "fedelissimi" aggiungiamo che gli "strani" comportamenti visti nel riquadro 3 dell'articolo suddetto possono ormai essere compresi, tenendo conto da un lato delle informazioni qui fornite, e dall'altro del fatto che, in sede di INPUT, l'interprete pone nel BUFFER Basic uno 0 al termine del dato immesso; tale 0 viene successivamente interpretato dall'esecutore come un fine-linea, e quindi l'esecutore procede come nelle fasi (III), (IV) eccetera.

Questo stesso fenomeno spiega anche il bug segnalato nell'articolo "La virgola è mobile", CCC N.27.

Linee indelebili e linee irriconoscibili

Se, una volta proceduto come indicato nel Riquadro 2, si vuole aggiungere un'ulteriore protezione al programma, si può eseguire il comando diretto:

POKE PEEK(43)+256*PEEK(44)+4, 250 (R)

Tale operazione "rinumera" la prima linea di programma assegnandole il numero 64000: eseguendo un LIST si otterrà:

64000 SYS 43256

seguito da SYNTAX ERROR; d'altronde la linea in questione è ora indelebile: provando a digitare:

64000 (return)

si otterrà solo SYNTAX ERROR (i numeri di linea ammessi vanno da 0 a 59999); ed inoltre anche il tentativo di far listare il resto del programma (digitando LIST 1-59999 anziché solo LIST) naufraga miseramente: incontrando la linea 60000 il computer ritiene di avere esaurito il suo compito, e non listerà nulla.

Tale tecnica ha due controindicazioni ma i rimedi sono semplici:

- **l'operazione di rinumerazione va effettuata solo se si è sicuri di non dover ulteriormente modificare il programma: quando è presente la linea 64000 il tentativo di correggere, ad esempio, la linea 1000 non modificherà affatto la vera linea 1000, ma ne costruirà un'altra che sarà inserita prima della linea 64000. Il rimedio ovvio è eseguire la rinumerazione solo a opera finita.**

- **se il programma presenta dei "GOTO all'indietro", come ad es.**

5500 GOTO 1000

in fase di esecuzione si otterrà il messaggio Undef'd Statement, seguito da ERROR IN 5500 ed il motivo è chiaro: cominciando la ricerca da inizio Basic, ed incontrando per prima la linea 64000, il computer riterrà che la linea 1000 non esiste (leggendo l'articolo si capisce perchè un tale problema non si pone nei "GOTO all'avanti").

Se il programma prevede rinvii all'indietro basterà inserire, nelle linee di disabilitazione 23456-23460, anche un'istruzione di ripristino del numero di linea iniziale:

POKE PEEK(43)+256*PEEK(44)+4,0

ed il gioco è fatto!

Conversione dec/esa (20953/21106)

a cura di Alessandro de Simone

Chiunque programmi in Linguaggio Macchina conoscerà certamente l'importanza della numerazione esadecimale.

Il codice riconosciuto dal computer, e che può contenere una locazione di memoria, è un insieme di otto cifre in base 2, vale a dire che gli unici valori emessi ed accettati dal microprocessore sono lo zero e l'uno, che vengono chiamati anche livelli logici, perchè corrispondono ad una particolare situazione elettronica di presenza (1) ovvero di assenza (0) di tensione.

Questi otto stati o livelli logici, combinandosi diversamente tra di loro, possono dare luogo a ben 256 combinazioni differenti. Supponendo di avere il codice binario seguente...:

01011011

...il valore è espresso in codice binario, esattamente allo stesso modo in cui noi scriviamo i nostri numeri in base 10. Quell'accozzaglia di uno e di zero corrisponde dunque a:

0*2 ↑ 7+
1*2 ↑ 6+
0*2 ↑ 5+
1*2 ↑ 4+
1*2 ↑ 3+
0*2 ↑ 2+
1*2 ↑ 1+
1*2 ↑ 0=
91

Non è difficile accorgersi della notevole complessità di un tale sistema di numerazione, almeno per l'uomo, che è abituato a ragionare in base 10 (chissà se avessimo avuto solo 8 dita che cosa sarebbe successo!).

Per ovviare all'inconveniente indicato, i matematici hanno pensato bene di dividere quella cifra così complessa in due parti (dette "nibble") attribuendo a ciascuna di esse un codice alfanumerico a seconda del numero indicato.

Ne segue che, ad esempio:

%0101 = $5 = ↑ 5
%1011 = $B = ↑ 11

in cui il simbolo di percentuale (%) indica un numero in notazione binaria, il simbolo del diesis (#) la notazione decimale e quello del dollaro ($) indica la numerazione esadecimale (a base 16).

In quest'ultima, oltre ai numeri, sono utilizzate le prime sei lettere dell'alfabeto e ciò per sopperire alla mancanza di adeguati simboli per indicare numeri superiori a 10. Questo fatto, anzichè complicare la situa-

```
1000 PRINTCHR$(147)"CONVERSIONE
     DECIMALE-ESADECIMALE":PRINT
1010 PRINT"ESEMPIO D'USO:":PRINT
1020 PRINT"SYS XXXX,#128":PRINT"
     SYS XXXX,$00FB":PRINT
1030 PRINT"VIENE VISUALIZZATO IL
      VALORE DECIMALE OPPURE ESA
     DECIMALE"
1040 RETURN
1050 DATA 032,115,000,201,035,24
     0,009,201,036,240
1060 DATA 063,162,022,076,055,16
     4,032,115,000,032
1070 DATA 138,173,032,247,183,16
     9,036,032,071,171
1080 DATA 160,001,185,020,000,07
     4,074,074,074,201
1090 DATA 010,144,003,024,105,00
     7,105,048,032,071
1100 DATA 171,185,020,000,041,01
     5,201,010,144,003
1110 DATA 024,105,007,105,048,03
     2,071,171,136,192
1120 DATA 000,016,215,096,160,00
     1,169,035,032,071
1130 DATA 171,032,115,000,056,23
     3,048,048,178,201
1140 DATA 010,144,010,233,017,04
     8,170,201,006,176
1150 DATA 166,105,010,010,010,01
     0,010,153,020,000
1160 DATA 032,115,000,056,233,04
     8,048,149,201,010
1170 DATA 144,010,233,017,048,14
     1,201,006,176,137
1180 DATA 105,010,025,020,000,15
     3,020,000,136,192
1190 DATA 000,016,194,166,020,16
     5,021,032,205,189
1200 DATA 032,115,000,096,-1,134
     78
```

zione, come potrebbe sembrare a prima vista, anzi la semplifica, perchè semplicemente accostando i due numeri ottenuti ($5 e $B, ad esempio) otteniamo la cifra esadecimale che corrisponde appunto al nostro #91 calcolato prima:

5*16 ↑ 1 + 11*16 ↑ 0 = 91

I vantaggi della numerazione esadecimale sono molteplici: innanzitutto qualsiasi valore riconosciuto dal microprocessore sarà espresso con due soli simboli (%11111111 = ↑255 = $FF). In secondo luogo, ma non meno importante, così facendo il programmatore ha sempre sott'occhio la situazione interna (e quindi binaria) della macchina, perchè con un po' di allenamento non è difficile imparare a memoria la tabellina di conversione da binario (con 4 sole cifre) ad esadecimale (fino ad "F"), per poi fare rapidamente tutte le conversioni di cui ha bisogno.

Naturalmente, alla conversione da decimale ad esadecimale e viceversa provvede la mini-routine presentata, con una sintassi veramente semplice:

SYS DECESA, ↑ numero decimale

oppure

SYS DECESA,$ numero esadecimale

Automaticamente, dunque, a seconda del simbolo che è anteposto al numero, il programma distingue le due basi numeriche del numero che segue e converte lo stesso nell'altra.

Importante sottolineare il fatto che il massimo valore accettato è ↑65535 e, naturalmente, $FFFF.

Per quanto riguarda i valori esadecimali, è necessario che vengano indicati tutti i 4 valori che compongono il codice a 16 bit, anche se corrispondono a zeri in posizioni non significative (non $FB dunque, ma $00FB).

Gli algoritmi utilizzati da questa routine sono alquanto interessanti, per cui cosigliamo vivamente di procedere ad una analisi dettagliata della stessa.

```
10 REM  DIMOSTRATIVO DEC-ESA
20 X=20953:REM  INDIRIZZO SUGG
   ERITO SU C.C.C.
30 PRINT"CONVERSIONE DECIMALE:
   #12356"
40 SYS20953,#12356
50 :
60 PRINT:PRINT"CONVERSIONE ESA
   DECIMALE:$C0FA"
70 SYS20953,$C0FA
```

Come utilizzare le routine

Sul N.31 di Commodore Computer Club è iniziata una nuova rubrica che ha lo scopo di venire incontro ai principianti (senza trascurare gli esperti), che desiderano potenziare al massimo le caratteristiche del proprio computer.

Il Basic presenta, infatti, carenze notevoli che possono esser limitate ricorrendo all'uso di routine in linguaggio macchina (LM): è sufficiente attenersi alle istruzioni pubblicate per utilizzare i sottoprogrammi LM con la massima semplicità. Gli "esperti" potranno fare a meno di seguire le istruzioni ed utilizzare direttamente i programmi L.M. pubblicati.

I principianti, invece, è opportuno che leggano con attenzione le "istruzioni per l'uso".

0' Se questa è la prima volta che leggete la rivista, accendete il vostro Commodore 64 e saltate al punto N.2.

1' Accendete il computer e, se desiderate "fondere" alcune (o tutte) le routine di questo numero con quelle tratte dai numeri precedenti (a patto, ovviamente che ne siate in possesso), caricate il file-programma "Nuovo Sistema" (nome standard adottato) con una delle due forme sintattiche che si riferiscono, rispettivamente, ai possessori di nastro o disco:

Load "Nuovo Sistema",1,1
Load "Nuovo Sistema",8,1

Subito dopo digitate NEW e premete il tasto Return.

2' Caricate il programma "Fissa Top di memoria" e lanciatelo col solito RUN. Alla domanda "Ultima locazione?" digitate 20000 e, alla successiva richiesta di conferma, premete il tasto "S". Le altre informazioni che appaiono sul video possono esser comprese solo dagli esperti: i principianti possono tranquillamente ignorarle e saltare alla prossima fase (N.3).

3' Caricate (o digitate dalla rivista) il programma "Caricatore".

4' Digitate dalla rivista la routine che interessa (scritta sempre in Basic, contenente in prevalenza istruzioni Data e numerata da 1000 in poi).

5' Effettuate una copia di sicurezza del programma che rappresenta la "fusione" dei due listati ("Caricatore" + routine Basic pubblicata).

6' Dopo aver digitato Run, alla domanda "Da quale locazione?" rispondete con l'indirizzo iniziale suggerito nello stesso titolo della routine in oggetto. Se il computer, dopo alcuni secondi, visualizza, come indirizzo finale, un valore diverso da quello pubblicato nel titolo (oppure il messaggio "Errore di trascrizione"), interrompete il lavoro (tasti Run Stop e Restore) e verificate con attenzione quanto avete trascritto da rivista.

Se, invece, compaiono messaggi "confortanti" (Routine allocata da... a... Attivare con Sys... ed altre informazioni comprensibili dagli esperti), digitate il programma dimostrativo e lanciatelo: da questo momento avete a disposizione una nuova routine LM da attivare mediante SYS come indicato nelle istruzioni pubblicate per ciascuna routine.

7' Ripetete le operazioni, dal punto 3 in poi, per ciascuna routine pubblicata che intendete

FISSA TOP MEMO

```
100 PRINTCHR$(147)"FISSA TOP DI
    MEMORIA"
110 INPUT "ULTIMA LOCAZIONE";X:
    X-X-1
112 PRINT:PRINT"I VALORI ATTUAL
    I SONO:":PRINT
113 X1-INT(X/256):X2-X-(X1*256)
115 PRINT"PEEK(55):"PEEK(55):PR
    INT"PEEK(56):"PEEK(56)
116 PRINT"FRE(0):"FRE(0)
117 PRINT:PRINT"I VALORI NUOVI
    SAREBBERO:":PRINT:PRINT"PEE
    K(55):"X2
118 PRINT"PEEK(56):"X1
120 PRINT:PRINT"CONFERMI? (S/N)
    "
130 IF PEEK(197)-64 THEN 130
135 IF PEEK(197)<>13 THEN POKE
    198,0: RUN
150 POKE 55,X2:POKE 56,X1: RUN1
    60
160 PRINT"FRE(0):"FRE(0):PRINT:
    PRINT"NEW"
```

SAVE ZONA RAM

```
150 PRINTCHR$(147):INPUT "LOCAZ
    IONE INIZIALE";X
160 INPUT "LOCAZIONE FINALE";T
170 PRINT:PRINT"INIZIO:"X:PRINT
    "FINE:"T
180 IF T<-X THEN RUN
190 PRINT:PRINT"CONFERMI? (S/N)
    "
200 GET A$:IF A$-"" THEN 200
210 IF A$<>"S" THEN RUN
220 PRINT:INPUT "NOME FILE";A$
230 PRINT"1- CASSETTA":PRINT"2-
    DISCO"
240 GET B$:IF B$-"" THEN 240
250 IF B$-"1" THEN W-1:GOTO 280
260 IF B$-"2" THEN W-8:GOTO 280
270 GOTO 240
280 PRINTCHR$(147);
290 POKE 198,5:POKE 631,19:POKE
    632,13:POKE 633,13:POKE 63
    4,13:POKE 635,0
300 X1-INT(X/256):X2-X-(X1*256)
    :PRINT"P⌐44,"X1":P⌐43,"X2;
310 Y1-INT(T/256):Y2-T-(Y1*256)
    :PRINT":P⌐46,"Y1":P⌐45,"Y2
320 PRINT:PRINT:PRINT"S*"CHR$(3
    4)A$CHR$(34)","W",1"
330 PRINT:PRINT:PRINT:PRINT:PRI
    NT"SYS64738"
```

CARICATORE

```
150 REM PER UTILIZZARLO, LEGGI
    LE ISTRUZIONI PUBBLICATE SU
160 REM COMMODORE COMPUTER CLUB
170 :
180 Y--1:GOSUB 1000:PRINT:INPUT
    "DA QUALE LOCAZIONE";X
190 READ W:Y-Y+1:IF W<0 THEN 21
    0
200 GOTO 190
210 PRINT"PRIMA LOCAZIONE -"X
220 PRINT"ULTIMA LOCAZIONE -"X+
    Y-1:PRINT
230 PRINT"CONFERMI? (S/N)"
240 GET A$:IF A$-"" THEN 240
250 IF A$-"S" THEN RESTORE :T-X
    :GOTO 280
260  RUN
270 :
280 GOSUB 1000:PRINT:PRINT"ATTE
    NDERE...":PRINT:W-0
290 READ B:IF B>-0 THEN POKE T,
    B:T-T+1:W-W+B:GOTO 290
300 READ B:IF B<>W THEN PRINT:P
    RINTCHR$(18)"ERRORE DI TRAS
    CRIZIONE":END
310 PRINT"ROUTINE ALLOCATA DA"X
    "A"T-1"COMPR.":PRINT
320 PRINT"ATTIVARE CON SYS"X:PR
    INT
330 X1-INT(X/256):X2-X-(X1*256)
    :PRINT"POKE44,"X1":POKE43,"
    X2;
340 Y1-INT(T/256):Y2-T-(Y1*256)
    :PRINT":POKE46,"Y1":POKE45,
    "Y2:END
```

"collezionare" non dimenticando di digitare NEW dopo ogni felice conclusione della fase N.6. Ai principianti consigliamo vivamente di trascriverle tutte in modo da aumentare la propria esperienza e, soprattutto, per evitare incomprensioni degli articoli che leggeranno su Commodore Computer Club.

8/Caricate, dopo un nuovo NEW, il programma "Save Zona Ram" e, dopo il Run, alla domanda "Da quale locazione?" rispondete con 20000. Alla seconda domanda "A quale locazione?" ripondete digitando l'indirizzo finale dell'ultima routine trascritta. A seconda se avete un registratore oppure un drive, sul nastro (oppure sul disco) vi ritroverete, dopo aver risposto alle varie domande, il file-programma "Nuovo Sistema" (nome che suggeriamo di assegnare quando compare la relativa domanda). Tale file-programma (da caricare come indicato al punto 1) sarà utilissimo sia per arricchire la vostra raccolta (trascrivendo le routine dei prossimi numeri di Commodore Computer Club), sia per utilizzarle in vostri listati.

9/Digitate SYS 64738 oppure premete il tasto di Reset (se lo possedete) in modo da rimettere "a posto" il computer. Caricate il programma "Fissa Top di memoria" e rispondete con 20000 alla domanda che vi porrà: da questo momento potete disporre sia delle consuete istruzioni Basic che delle routine LM richiamabili con le corrispondenti SYS. Non dimenticate di ripetere la presente fase (N.9) tutte le volte che premete il tasto di Reset o dopo un reset software (SYS 64738). Se, invece, spegnete il computer, sarà necessario attuare la fase N.1 e N.2 per inserire nuovamente nel calcolatore le nuove routine ed usarle senza pericolo.

Collaborazione dei lettori

I lettori che intendono collaborare devono inviare (almeno) tre routine, relativi listati dimostrativi ed articoli esplicativi. Le norme da seguire per la stesura dei listati (piuttosto rigide, per ovvi motivi di compatibilità) sono state segnalate sul N.31. Per ulteriori informazioni, comunque, è possibile telefonare in Redazione (02/8467348) chiedendo di Michele Maggi.

Locate cursor (21107/21156)

Nel Basic del Commodore 64 manca stranamente una particolare istruzione, molto importante per la gestione della grafica nello schermo in bassa risoluzione. Alludiamo alla famigerata PRINT AT, che permette di situare il cursore sullo schermo alla posizione indicata dai due parametri specificati dall'utente.

Sfruttando opportunamente questa routine si possono creare divertenti videate grafiche senza ricorrere all'alta risoluzione, oppure creare tabelle, grafici ed addirittura studi di funzioni con una risoluzione di 25 per 40 caratteri, spesso più che sufficienti per i nostri scopi.

In pratica, la routine in Linguaggio Macchina che viene presentata è molto semplice (ed anche molto rapida da digitare!).

Ci permettiamo di suggerire al lettore, in caso di utilizzo in programma Basic, di specificare all'inizio una variabile il cui nome sia oculatamente scelto con l'indirizzo di partenza della stessa routine, in modo che poi la successiva interpretazione del programma in questione non sia irrimediabilmente ostacolata da stranissime SYS.

Per esempio, supponendo di rilocare la routine a partire da 21107 (come da noi suggerito), si potrebbe inizializzare il programma con:

LOCATE=21107

La sintassi di questa nuova "istruzione" è molto semplice, non ricorrendo a nessuna POKE preliminare. Bisogna solo richiamare la routine, facendo seguire il tutto dai valori della posizione del cursore, separati da virgole.

Volendo posizionare il cursore nell'angolo in alto a sinistra dello schermo (0,0), non dovremo fare altro che digitare:
SYS LOCATE,0,0

I due parametri riguardano rispettivamente la posizione orizzontale e quella verticale, o meglio le coordinate in ascissa e quelle in ordinata.

Fatto abbastanza interessante, i parametri possono essere specificati anche con variabili, quindi la flessibilità nell'uso risulta notevole.

```
1000 PRINTCHR$(147)"LOCATE CURSO
     R (PRINT AT)":PRINT
1010 PRINT"SYS XXXX,X,Y":PRINT
1020 PRINT"POSIZIONA IL CURSORE
     SU VIDEO IN BASSA RISOLUZIO
     NE"
1030 PRINT:PRINT"0<X<24":PRINT"0
     <Y<39"
1040 RETURN
1050 DATA 032,115,000,032,138,17
     3,032,247,183,165
1060 DATA 021,208,032,165,020,07
     2,032,253,174,032
1070 DATA 138,173,032,247,183,16
     5,021,208,016,166
1080 DATA 020,104,168,192,040,17
     6,008,224,025,176
1090 DATA 004,032,240,255,096,16
     2,015,076,055,164
1100 DATA -1,5707
```

```
10 PRINTCHR$(147):PRINT"DIMOST
   RATIVO: LOCATE CURSOR"
20 X=21107:REM  INDIRIZZO SUGG
   ERITO SU C.C.C.
30 SYS21107,12,7:PRINT"ECCO QU
   I IL CURSORE"
```

Beep (21157/21260)

Questa routine consente di produrre un "Beep" sonoro di durata e frequenza desiderate. Grazie al potentissimo SID (Sound Interface Device) contenuto nel Commodore 64, generare suoni risulta semplice, anche se non semplicissimo.

Come tutti certamente sapranno, per produrre una determinata nota bisogna selezionare la forma d'onda, il volume, l'ADSR (Attak, Decay, Sustain, Release), i filtri. Il tutto tramite POKE, che non sempre risultano di immediata comprensione.

Per ovviare a questo inconveniente, e ritenendo che non sempre è indispensabile assegnare sofisticate specifiche ad un segnale acustico, abbiamo pensato di creare una routine che, molto brutalmente, genera un beep.

E' comunque possibile controllarne la frequenza tramite un parametro che viene "passato" con la chiamata, mentre tramite un altro si determina la durata della nota generata.

Anche in questo caso consigliamo

di attribuire ad una variabile la locazione di inizio della routine, per evitare confusioni in programmi Basic che la utilizzano. La sintassi è la seguente:

SYS BEEP,Durata,Freq.

La durata è espressa in decimi di secondo, mentre per ottenere la frequenza esatta in Hertz bisogna moltiplicare il parametro della frequenza per una costante:

Hertz=0.06097*Freq

Sia la durata che la frequenza possono arrivare ad un valore massimo di 65535.

Nel caso si volesse disassemblare il programmino, è opportuno avere a portata di mano la Guida al Sistema Operativo, dal momento che si fa uso abbondante di routine della ROM, per semplificare e ridurre drasticamente il numero di istruzioni.

```
1000 PRINTCHR$(147)"BEEP: ESEMPI
     O D'USO":PRINT
1010 PRINT"SYS XXXX,Y,Z":PRINT
1020 PRINT"Y=DURATA SUONO IN DEC
     IMI DI SEC."
1030 PRINT"Z=FREQUENZA (HZ)*0.06
     097":PRINT
1040 PRINT"0<Y<65535":PRINT"0<Z<
     65535"
1050 RETURN
1060 DATA 169,000,141,004,212,14
     1,011,212,141,018
1070 DATA 212,169,015,141,024,21
     2,169,008,141,023
1080 DATA 212,169,000,141,005,21
     2,169,240,141,006
1090 DATA 212,032,115,000,032,13
     8,173,032,247,183
1100 DATA 165,020,072,165,021,07
     2,032,253,174,032
1110 DATA 138,173,032,247,183,10
     4,133,096,104,133
1120 DATA 095,005,096,240,033,16
     5,020,166,021,141
1130 DATA 000,212,141,001,212,16
     9,017,141,004,212
1140 DATA 032,179,238,165,095,19
     8,095,170,208,246
1150 DATA 198,096,165,096,201,25
     5,208,238,169,000
1160 DATA 141,004,212,096,-1,127
     67
```

```
10 PRINTCHR$(147)"DIMOSTRATIVO
    BEEP":PRINT
20 X=21157:REM  INDIRIZZO SUGG
   ERITO SU C.C.C.
30 SYS21157,200,300
40 PRINT"DURATA="200"FREQUENZA
   ="300"
```

Mappa della memoria di Nuovo Sistema

(Elenco delle routine pubblicate)

Il primo valore indica l'indirizzo di partenza (coincidente con la SYS da impartire), mentre, il secondo, l'ultima locazione contenente l'ultimo dato.
Il numero fra parentesi, invece, si riferisce al numero di C.C.C. in cui sono state pubblicate le routine stesse.

20000/20011 GoTo Calcolato (31)
20012/20049 GoSub Calcolato (31)
20050/20128 Interp A$ (31)
20129/20188 Cambia colore (31)
20189/20245 Scroll Carattere (31)
20246/20302 Cancella caratt. (31)
20303/20445 GoSub Label (32)
20446/20562 GoTo Label (32)
20563/20596 Restore linea (33)
20597/20682 Disk Tool (33)
20683/20775 Directory (33)
20776/20858 Scroll Flag (34)
20859/20914 Deek (34)
20915/20952 Doke (34)

(Le routine di questo numero dono opera di Simone Bettola)

Il "vero" Linguaggio Macchina

Un nuovo approccio al cervello del popolare computer

Di solito chi non si accontenta più del Basic e decide di studiare un linguaggio di più vasta portata, si trova inevitabilmente attratto dal linguaggio macchina dal momento che è immediatamente disponibile grazie alle stesse istruzioni Basic PEEK, POKE e SYS.

Sul mercato, del resto, si trovano numerosi Tool che vanno considerati alla stregua di "filtri" tra l'utilizzatore ed il microprocessore allo scopo di facilitare la programmazione del calcolatore.

Questi Tool, di norma, si dividono in due categorie fondamentali:

• Semplici Monitor, vale a dire programmi che visualizzano, quasi sempre in forma esadecimale, il contenuto della memoria e consentono modifiche, trascrizioni di blocchi, registrazioni, lanci eccetera.

• Assembler / Disassembler che consentono di programmare utilizzando codici mnemonici di presunta facile interpretazione.

I più moderni Tool contengono sia Monitor che sezioni Assembler.

Ma è proprio vero che questi Tool aiutano il programmatore? Siamo realmente convinti che un microprocessore può esser programmato solo in codice esadecimale oppure ricorrendo ai codici mnemonici?

La risposta, ovviamente, è negativa per Armando Caiazzo, autore del volume "Il vero Linguaggio Macchina del Commodore 64".

Sua opinione è che, per imparare realmente la vera lingua del computer, è sufficiente limitarsi a conoscere i 151 codici macchina scritti, tra l'altro, ricorrendo alla consueta notazione decimale.

In effetti, nelle quasi 400 pagine del volume citato, non compare una sola volta la "traduzione" esadecimale di un codice (tranne che per paragoni in varie tabelle) nè viene proposta la digitazione di un Tool, nè si ricorre a codici mnemonici.

Il libro, dunque, affronta l'argomento ricorrendo alle sole istruzioni Read...Data per introdurre i codici macchina. Centinaia sono gli esempi riportati con applicazioni a tutte le potenzialità del popolare computer: visualizzazione di messaggi su schermo, musica, sprite eccetera. Ogni nuova istruzione affrontata viene immediatamente tradotta in mini programma di facilissima digitazione.

Le appendici, manco a dirlo, continuano ad utilizzare esclusivamente il codice decimale per illustrare in dettaglio le 151 istruzioni, le routine delle Rom, le tecniche di Interrupt, l'uso dei registri di stato e tutto ciò, insomma, che riguarda da vicino il microprocessore del Commodore 64.

Un volume, quindi, che affronta in modo originale e unico un argomento che tanto interesse suscita nell'utente che, stanco del Basic, desidera dedicarsi a qualcosa di più impegnativo senza impegnarsi... troppo!

A. Caiazzo
Il vero Linguaggio Macchina
del Commodore 64
Soc.Ed. "Linguaggio Macchina"
C.so Garibaldi, 95
82100 Benevento
L.30000

Programma: Uomo

Un'avventura appassionante, della nota collana "Urania", che si svolge nel mondo dei computer

di Alessandro de Simone

Mi è piaciuto questo libro di fantascienza che ho letto nei momenti di ozio estivo (è uscito in edicola, infatti, a metà agosto).

Gli autori dimostrano di avere una notevole fantasia che tuttavia non eccede nelle solite esagerazioni tipiche degli scrittori di "Science fiction".

Sicuramente hanno letto qualcosa di psicologia (soprattutto per ciò che riguarda il mondo dei sogni), posseggono (o hanno visto realmente funzionare) un personal computer, conoscono i problemi delle protezioni del software e sono aggiornati sulle possibilità future di collegamenti tra cervelli elettronici e cervelli umani. Per quanto riguarda quest'ultimo argomento, che al giorno d'oggi è ancora a livelli poco più che teorici, gli autori, dignitosamente, preferiscono sfociare nella fervida fantasia piuttosto che rendere credibili apparecchiature che non è ancora possibile prevedere nel loro aspetto finale (se mai verranno costruite!).

Per quanto riguarda il resto, la vicenda, recante data 1999, si snoda avendo come cornice strumenti informatici che già sono attualmente disponibili (collegamenti tra computer, reti locali e remote, controlli computerizzati anche mediante terminali portatili) o, comunque, "credibili", come camion e convogli ferroviari privi di conducenti "umani" e guidati esclusivamente da calcolatori.

Pur se di sfuggita, il lettore è invitato a soffermarsi sull'impiego legale delle banche dati sui cittadini, problema che, attualmente, è dibattuto perfino in sede politica e sindacale.

Per ciò che riguarda la trama, diremo solo che il protagonista del romanzo un bel giorno si sveglia con un vuoto nella memoria che vuole, ovviamente, colmare al più presto; nelle sue indagini si accorge di possedere una singolare prerogativa: se si trova nei pressi di un calcolatore, è in grado di introdursi mentalmente nel suo "bus" e rendersi conto delle operazioni che compie.

Non continuiamo, ovviamente, nella descrizione anche perchè il romanzo si tinge di giallo con consueta sorpresa finale.

Siamo sicuri che i lettori trovino quantomeno "simpatica" l'iniziativa di affrontare temi che, pur non trattando argomenti strettamente legati ai computer Commodore, parlino comunque del mondo dell'informatica nel suo significato più vasto che comprende, diamine!, anche momenti di pausa...

R.Zelazny F.Saberhagen
"Programma: Uomo"
Urania N.1029
Mondadori Editore
L.3000

Enciclopedia di routine

a cura di Alessandro de Simone

14900 Cancella finestre schermo

(Commodore 64)

Questa routine cancella una porzione rettangolare, di dimensioni prefissate, in una qualunque zona dello schermo (che si presuppone nella posizione "default" a partire da 1024).

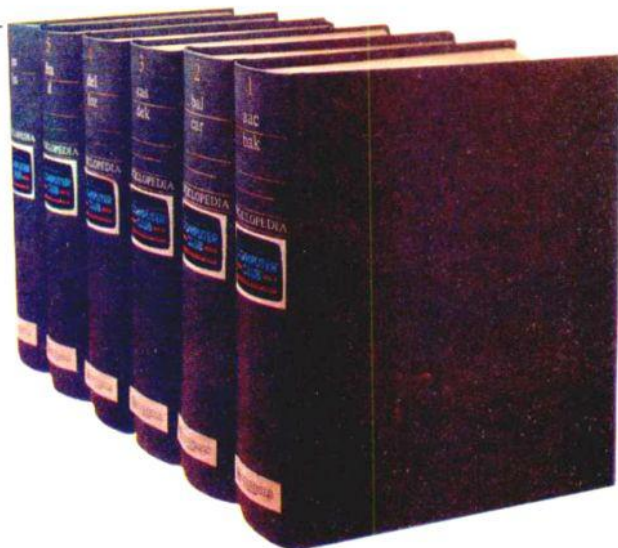
Prima di richiamarla devono essere assegnati alle variabili X1 e X2, rispettivamente, il valore-schermo dell'angolo superiore sinistro della finestra e quello dell'angolo inferiore destro: in tal modo viene determinata sia la dimensione della finestra che la sua posizione.

Ad esempio: X1 = 1401, X2 = 1606 cancella una finestra quadrata al centro dello schermo; X1 = 1024, X2 = 1503 cancella l'intera metà superiore, e così via.

Se i due valori non sono coerenti (se il secondo non si trova a destra e in basso rispetto al primo) la routine non viene eseguita e in X0$ sarà contenuto il messaggio "ERR".

Il funzionamento è basato sulla creazione e stampa ripetuta di una stringa formata da un adeguato numero di spazi, formattando la stampa in funzione dei parametri assegnati all'inizio.

Naturalmente, se in seguito volete scrivere nella finestra così creata, dovrete usare un'opportuna routine di posizionamento del cursore.

```
 100 REM ESEMPIO D'USO
 110 REM CANCELLA UNA FINESTRA
 120 REM IN QUALUNQUE PUNTO DELL
     O SCHERMO
 130 REM SOLO PER C-64
 135 :
 140 PRINTCHR$(147)"VALORI LOCAZ
     IONI SCHERMO
 150 PRINT"(ANGOLI: SUPERIORE SI
     NISTRO, INFERIORE DESTRO)
 160 PRINT:PRINT:INPUT X1,X2
 170 PRINTCHR$(147);:FOR X=1 TO
     999:PRINT"X";:NEXT
 180 GOSUB 14900
 190 GET A$:IF A$="" THEN 190
 200 GOTO 140
 210 :
9999 END
```

```
14900 X0$="":X1$="":FOR X3=1 TO 2
      4:X1$=X1$+CHR$(17):NEXT:X1$
      =CHR$(19)+X1$
14905 IF X1<1024 OR X1>1983 OR X2
      <1024 OR X2>1983 THEN X0$="
      ERR":RETURN
14910 X3=INT((X1-1024)/40):Y3=X1-
      40*X3-1024
14915 X4=INT((X2-1024)/40):Y4=X2-
      40*X4-1024
14920 IF X3>X4 OR Y3>Y4 THEN X0$=
      "ERR":RETURN
14925 X$="":FOR X=0 TO (Y4-Y3):X$
      =X$+" ":NEXT
14930 FOR X=X3 TO X4:PRINTLEFT$(X
      1$,X+1)SPC(Y3)X$:NEXT:RETUR
      N
14990 REM VARIABILI: X,X1,X2,X3,X
      4,Y3,Y4,X$
14992 REM CANCELLA UNA PORZIONE
14994 REM RETTANGOLARE DELLO SCHE
      RMO-
14996 REM X1=ANGOLO SUPERIORE SIN
      ISTRO
14998 REM X2=ANGOLO INFERIORE DES
      TRO
14999 REM NOME: CANCELLA FINESTRE
       SCHERMO
```

15000 Frammenta-schermo

(Commodore 64)

Questa routine in l.m. propone, come elegante alternativa al solito "Clear", una frammentazione dello schermo in tanti quadratini colorati.

La routine "spara" sullo schermo, in maniera casua-

le, degli spazi-reverse colorati, per una durata di circa due secondi (di regola sufficienti a riempire lo schermo), e viene conclusa da un normale "Clear".

Se la zona di memoria suggerita (680-755) fosse utilizzata diversamente, la routine può essere rilocata semplicemente ponendo la nuova locazione di inizio al posto del "680" nella riga 15045, e modificando i relativi SYS.

```
  100 REM ESEMPIO D'USO
  110 REM FRAMMENTA-SCHERMO
  120 REM SOLO PER C-64
  125 :
  126 REM BY ROBERTO MORASSI
  127 :
  130 PRINTCHR$(147)"PRIMA SCHERM
      ATA (NERO)
  140 GOSUB 15000:SYS680,0
  150 PRINT"SECONDA SCHERMATA (RO
      SSO)":FOR X=1 TO 2E3:NEXT
  160 SYS680,2
  170 PRINT"TERZA SCHERMATA (GIAL
      LO)":FOR X=1 TO 2E3:NEXT
  180 SYS680,7
  190 PRINT"....ECCETERA !
  200 :
 9999 END
15000 X9$="03224118316925514101 52
      121691281410182121690000322
      192551730272121 33"
15010 X9$=X9$+"251173027212074074
      074074074074009004133252160
      000169160145251"
15020 X9$=X9$+"165252024105212133
      252138145251234234234234234
      234234234036162"
15030 X9$=X9$+"016210140015212140
      018212169147076210255"
15035 X7=0:X8=0
15040 FOR X=1 TO LEN(X9$) STEP 3:
      X9=VAL(MID$(X9$,X,3))
15045 POKE 680+X8,X9:X8=X8+1:X7=X
      7+X9:NEXT
15050 IF X7<>11093 THEN PRINT"ERR
      ORE":END
15055 RETURN
15090 REM VARIABILI: X,X9$,X7,X8,
      X9
15092 REM ESEGUE IL CLEAR DELLO S
      CHERMO
15094 REM IN FRAMMENTI COLORATI
15099 REM NOME: FRAMMENTA-SCHERMO
```

15100 Lampeggio righe schermo

Questa brevissima routine fa lampeggiare sullo schermo messaggi di una o più righe, oppure, alternativamente (in controfase), righe diverse, o addirittura parti diverse di una medesima riga.

La frequenza del lampeggio viene prefissata assegnando alla variabile X1 valori da 0 a 7: più basso è il valore, più rapido è il lampeggio.

La routine viene richiamata subito prima di stampare il messaggio. Essa definisce alcune variabili-stringa (X1S, X2$, X3S, X4$) che chiamerei "variabili flip-flop": il loro valore,, infatti, dipende dal contenuto istantaneo del registro inferiore del clock (162).

Se, per esempio, X1 èfissata a 4, X1$ sarà uguale a "Rvs" se il contenuto di tale registro diviso 2 alla quarta è dispari, a "Rvoff", invece, se è pari.

X2$ assume gli stessi valori, ma invertiti.

X3$ e X4$, analogamente, saranno posti uguali a "bianco" "blu" o viceversa.

Queste variabili vengono inserite nel messaggio e ne formattano la stampa, alternativamente, in "rvs" "rvoff" oppure "bianco" "blu" (se quest'ultimo è il colore del fondo, il messaggio scompare).

Il GET di attesa ripete la stampa purchè si abbia l'accortezza di rimandare ogni volta il cursore nella posizione iniziale: il modo più semplice è quello di aggiungere all'ultima riga tanti "cursor-up" (CHR$(157)) quante sono le righe da ripetere.

La fantasia potrà suggerire molte altre varianti e applicazioni delle variabili flip-flop.

```
  100 REM ESEMPIO D'USO
  110 REM MESSAGGI LAMPEGGIANTI
  120 REM QUALSIASI COMPUTER
  130 PRINTCHR$(147);:INPUT "FREQ
      UENZA (0-7)";X1
  140 PRINT:PRINT
  150 GOSUB 15100:PRINTX1$"LAMPEG
      GIO NORMALE/REVERSE"
  160 PRINTX1$"LO STESSO"X2$" ALT
      ERNATO"CHR$(145)CHR$(145)
  170 GET A$:IF A$="" THEN 150
  180 PRINT:PRINT:PRINT:PRINT
  190 GOSUB 15100:PRINTX3$"LAMPEG
      GIO APPARE/SCOMPARE"
  200 PRINTX3$"LO STESSO"X4$" ALT
      ERNATO"CHR$(145)CHR$(145)
  210 GET A$:IF A$="" THEN 190
  220 PRINT:PRINT:PRINT"FINE DEMO
      "
  230 :
 9999 END
15100 X1$=CHR$(18+128*((PEEK(162)
      /2↑X1) AND 1))
```

```
15110 X2$=CHR$(146-128*((PEEK(162
      )/2↑X1) AND 1))
15120 X3$=CHR$(5+26*((PEEK(162)/2
      ↑X1) AND 1))
15130 X4$=CHR$(31-26*((PEEK(162)/
      2↑X1) AND 1))
15140 RETURN
15190 REM VARIABILI:X1,X1$,X2$,X3
      $,X4$
15192 REM FA LAMPEGGIARE I MESSAG
      GI
15194 REM SULLO SCHERMO, IN DUE M
      ODI
15196 REM DIVERSI
15198 REM X1=FREQUENZA LAMPEGGIO
      (0-7)
15199 REM LAMPEGGIO RIGHE SCHERMO
```

50100 Esame directory del disco

Sul N.26 di C.C.C. è già stata pubblicata una routine, in l.m. che svolge la stessa funzione.

La riproponiamo, in puro Basic, per consentire ai lettori di studiarla meglio.

La brevissima routine non ha bisogno di commenti: è sufficiente richiamarla per avere sullo schermo la directory del disco presente in quel momento nel drive, senza interferire in alcun modo con il programma in memoria.

Tenete presente che all'inizio della routine (righe 50100-50110) vengono chiusi tutti i file che fossero eventualmente aperti, onde evitare un "File Open Error".

```
100 REM ESEMPIO D'USO
110 REM DIRECTORY DEL DISCO
120 REM QUALSIASI COMPUTER
130 :
150 PRINTCHR$(147);
160 PRINT"INSERISCI UN DISCO,
170 PRINT"POI PREMI UN TASTO
180 GET A$:IF A$="" THEN 180
190 PRINTCHR$(147)"DIRECTORY DE
    L DISCO ATTUALE:":GOSUB 501
    00:GOTO 160
200 :
9999 END
50100 IF PEEK(152)=0 THEN 50115
50105 FOR X=0 TO 9:Y=PEEK(601+X):
      IF Y THEN CLOSE Y
50110 NEXT
50115 OPEN 15,8,15,"I":OPEN 1,8,0
      ,"$":Y$=CHR$(0)
50120 GET #1,X$,X$
50125 GET #1,X$,X$,X1$,X2$:X=ST:I
      F X=0 THEN PRINTASC(X1$+Y$)
      +256*ASC(X2$+Y$);
50130 IF ST THEN PRINT:CLOSE 1:CL
      OSE 15:RETURN
50140 GET #1,X$:IF X$="" THEN PRI
      NT:GOTO 50125
50150 PRINTX$;:GOTO 50140
50190 REM VARIABILI: X$,Y$,X1$,X2
      $,X,Y
50192 REM LEGGE E VISUALIZZA
50194 REM IL DIRECTORY DEL DISCO
50196 REM SENZA SOVRAPPORSI
50198 REM AL PROGRAMMA IN MEMORIA
50199 REM DIRECTORY DEL DISCO
```

Roberto Morassi

Elenco delle routine pubblicate

63941 REM 14800 SOSTITUISCE STRINGHE (34)
63942 REM 14700 SLITTA STRINGHE (34)
63943 REM 14600 RUOTA STRINGHE (34)
63944 REM 10500 INPUT PROGRAMMABILE (34)
63945 REM 14500 SCROLL SOLO TESTO (33)
63946 REM 14400 SPRITE MULTIUSO (33)
63947 REM 14300 ZOOM ESADECIMALE (33)
63948 REM 14200 VIDEO OROLOGIO (33)
63949 REM 11100 FUNZIONI INVERSE (32)
63950 REM 13200 CENTRATRICE MESSAGGI (32)
63951 REM 14100 FINESTRE DI TESTO (32)
63952 REM 14000 GESTIONE NOME DISCO (32)
63953 REM 13900 CARICA/SALVA PAG.VIDEO (31)
63954 REM 13800 MESSAGGI IN E.B.C.M. (31)
63955 REM 13700 BIT IMAGE MPS/803 (31)
63956 REM 13600 OR ESCLUSIVO (31)
63957 REM 13500 COMANDI FUORI PROGRAMMA (31)
63958 REM 13400 LINEE BASSA RISOLUZIONE (31)
63959 REM 13300 ELABORAZIONE STRINGHE (31)
63960 REM 13200 CENTRATURA FRASE (32)
63961 REM 13100 SCELTA MENU JOYSTICK (30)
63962 REM 13000 SCELTA MENU CURSORE (30)
63963 REM 12900 SCRITTA LAMPEGGIANTE (29)
63964 REM 12800 BORDO VIDEO TECHNICOLOR (29)
63965 REM 12700 FILL MEMORIA RAM (29)
63966 REM 12600 TEXT COPY (MPS 803) (29)
63967 REM 12500 CAMBIA COLORE PAG.TESTO (29)
63968 REM 12400 PRINT USING (31)
63968 REM 12400 PRINT USING (29)
63969 REM 12300 M.C.D. e m.c.m. (29)
63970 REM 50500 VISUALIZZA FILE (28)
63971 REM 50400 LEGGE FILE RELATIVI (28)
63972 REM 50300 SCRIVE SU FILE RELATIVI (28)
63973 REM 50200 CREA FILE RELAIVI (28)
63974 REM 50000 LEGGE BLOCCHI LIBERI (28)
63975 REM 12200 NUMERI CONGRUI (28)
63976 REM 12100 PROTEZIONE SOFTWARE (28)
63977 REM 12000 KOALA (27)
63978 REM 11900 SCAMBIA PAGINA VIDEO (27)
63979 REM 11800 SALVA RAM (27)
63980 REM 11700 CALCOLATRICE (27)
63981 REM 11600 SCOMPOSIZ.SILLABE (27)
63982 REM 11500 CAR.HI-RES (27)
63983 REM 11400 ISTOGRAMMI (27)
63984 REM 50100 ESAME DIRECTORY (26)
63985 REM 11300 FUNZ.INV.IPERBOLICHE (26)
63986 REM 11200 FUNZ.INV. TRIGONOM. (26)
63987 REM 11100 FUNZIONI INVERSE (26)
63988 REM 11000 FUNZIONI IPERBOLICHE (26)
63989 REM 10900 CONVERSIONE DEC-ESA (26)
63990 REM 10800 CONTROLLO DATA (25)
63991 REM 10700 IMPULSI SONORI (25)
63992 REM 10600 REVERSE SCHERMO (25)
63993 REM 10500 INPUT CONTROLLATO (25)
63994 REM 10400 INCOLONNAMENTO VIRGOLA (25)
63995 REM 50000 N. BLOCKS FREE(DISCO) (24)
63996 REM 10300 INPUT & CONTR/DEFAULT (24)
63997 REM 10200 ESTRAZ.PAROLA DA FRASE (24)
63998 REM 10100 CAMBIA COL.BORDO/FONDO (24)
63999 REM 10000 CORNICE POLICROMA (24)

Autoboot per C/128

Come mandare in esecuzione un programma presente su disco senza Load nè Run

di M.Maggi e U.Colapicchioni

Quando si inserisce nel drive il dischetto del CP/M, incluso nella confezione del C/128, inizia automaticamente a girare caricando e mandando in esecuzione il programma stesso.

Ci siamo chiesti se questa caratteristica (caricamento e lancio automatici) fosse esclusiva del CP/M oppure potesse essere sfruttata anche con nostri programmi.

Siamo andati a spulciare tutti i manuali che normalmente vengono forniti dalla Commodore, ma non abbiamo ovviamente (!) trovato niente che facesse al caso nostro.

Come spesso succede in questi casi, la sperimentazione è l'unica maniera per scoprire informazioni particolari non riportate sui manuali.

Siamo quindi andati a curiosare sulle tracce e sui settori del disco del CP/M con un analizzatore di tracce del C-64 che permette di visualizzare il contenuto dei settori del disco.

La prima cosa che abbiamo notato è che sul primo settore del disco è presente la sequenza dei caratteri "CBM", seguita da alcuni numeri.

Come qualche lettore ricorderà, nel C-64 il sistema di autostart delle cartridge si basa sull'uso della stessa sequenza, ovviamente allocata in memoria Rom e non su disco.

Abbiamo però pensato che fosse tradizione della Commodore usare le proprie iniziali per segnalare al sistema la presenza di un autostart e, supportati da tale convinzione, ne abbiamo poi avuto conferma.

La comodità di un caricamento automatico (autoboot) è evidente: affidando il programma ad una persona inesperta, la mettiamo in condizione di utilizzarlo senza doversi preoccupare di "trafficare" con directory e/o vari comandi di caricamento, con annesse possibilità di errori che, per un nuovo utente del computer, possono trasformarsi in difficoltà insormontabili.

Chiaramente la possibilità di far partire automaticamente i programmi potrà far comodo anche all'utente esperto, che aggiungerà in tal modo un tocco di professionalità al proprio software.

Spieghiamo qui di seguito il funzionamento di questo particolare autostart.

Come funziona AutoBoot

All'accensione del computer, il Sistema Operativo (S.O.) legge i byte contenuti nel primo settore del disco (Traccia 1 Settore 0) per controllare se è presente, c meno, la sequenza dei caratteri "CBM".

In caso negativo (drive spento o dischetto non predisposto) il sistema esegue le normali operazioni riguardanti l'inizializzazione standard.

Questo è il motivo del "ritardo" tra l'accensione del C/128 e la comparsa della scritta "READY".

Se, invece, sul dischetto si trova la sequenza, il controllo passa ad una routine che automaticamente carica ed esegue il programma specificato sul settore posto di seguito alla stessa sequenza.

Per utilizzare questa caratteristica dobbiamo predisporre un dischetto in modo da fargli contenere i dati necessari al Boot.

Il programmino che presentiamo serve appunto a questo scopo.

Come utilizzare il programma

Salvate il programma di queste pagine su un dischetto, formattatene uno nuovo per mezzo del comando HEADER "Nome Disco",ID (o analoghi) e, a formattazione avvenuta, date il RUN al programma.

Il drive inizierà a girare e in pochi secondi il dischetto sarà predisposto per l'autostart.

Se infatti richiedete la directory dopo le operazioni descritte, potrete notare che, rispetto ad un disco normalmente formattato, manca un blocco, che è stato occupato, appunto, dalla sequenza necessaria all'autostart.

Adesso caricate un qualsiasi vostro programma e salvatelo sotto il nome di "LOADER" (e solo con questo nome!) sul disco "trattato". E' intuitivo che se volete assegnare un nome diverso al programma da lanciare, sarà necessario cambiare, nel listato, il DATA corrispondente (vedi ultime righe del programma di queste pagine).

Per provare che tutto è in ordine, spegnete il computer e riaccendetelo in modo, ovviamente, 128.

Se avete seguito le semplici istruzioni, il programma verrà caricato e partirà da solo.

Se, invece, volete evitare il lancio automatico, sarà sufficiente mantenere il drive vuoto al momento dell'accensione (o del Reset).

Nel digitare righe di programma basic che contengono istruzioni DATA, è piuttosto facile incorrere in errori di digitazione. Supponiamo che un'ipotetica linea basic numerata con 1200 debba contenere i tre valori: 123, 456, 789. Ecco alcuni esempi di errori più frequentemente commessi:

1200 DATA,123,456,789

C'è una virgola dopo la parola "DATA". I dati letti dal computer sono, in questo caso, quattro: 0, 123, 456, 789. Se, infatti, non figura alcun carattere dopo l'istruzione DATA, automaticamente viene assunto il valore nullo (0).

1200 DATA 123,456,789,

In questo caso, dopo il numero 789, il computer, grazie alla presenza della virgola erroneamente inserita, "crede" che ci sia un altro valore e, non trovandolo, lo assume come nullo (0).

1200 DATA 1234,56,789

La virgola è posizionata male, vale a dire dopo il carattere "4" e non dopo il carattere "3". Il computer non può sapere se il valore esatto è 123 oppure 1234 e individuare un errore, in questo caso, risulta piuttosto laborioso.

```
100 REM  AUTOSTART C/128
110 REM  E DRIVE 1541 / 1571
120 REM  BY U.COLAPICCHIONI
130 REM  E M.MAGGI
140 :
150 PRINTCHR$(147)"PROGRAMMA DI
     AUTOSTART PER C-128":PRINT
160 PRINT"ATTENZIONE: INSERISCI
     UN DISCO FORMATTATO E PREM
    I UN TASTO"
170 GET A$:IF A$="" THEN 170
180 PRINT:PRINT"STO INIZIALIZZA
    NDO IL DISCHETTO...."
190 H$="CBM"+CHR$(0)+CHR$(0)+CH
    R$(0)+CHR$(0)+CHR$(0)+CHR$(
    0)
200 FOR X=1 TO 14:READ A$:H$=H$
    +CHR$( DEC (A$)):NEXT
210 READ A$,B$:H$=H$+A$+CHR$(34
    )+B$+CHR$(13)
220 OPEN 15,8,15:OPEN 2,8,2,"#"
230 PRINT#15,"B-P";2;0:PRINT#2,
    H$
240 PRINT#15,"B-A";0;1;0
250 PRINT#15,"U2";2;0;1;0:CLOSE
     2:CLOSE 15
260 PRINT:PRINT:PRINT"DISCHETTO
     PRONTO."
270 DATA A2,0A,86,D0,BD,16,0B
280 DATA 9D,49,03,CA,D0,F7,60
290 DATA "R,"
300 DATA "LOADER"
```

12 - Lire 8.000
Commodore Club - Dir. Resp. A. Ronchetti Edizioni Systems Editoriale Srl - V.le Famagosta 75 - 20142 Milano - Reg. Trib. MI. n. 104 del 25/2/84 - Distr. MePe.

Registrate il mio abbonamento annuale a Commodore Computer Club.

☐ Ho versato oggi stesso il canone di L. 35.000 a mezzo c/c postale n°37952207 intestato a:
Systems Editoriale - V.le Famagosta, 75 - 20142 Milano

☐ Ho inviato oggi stesso assegno bancario n.. ..
per l'importo di L. 35.000 intestato a Systems Editoriale
Si prega di scrivere il proprio nome e l'indirizzo completo in modo chiaro e leggibile. Inviare la fotocopia del bollettino di c/c postale.

Considerando che i numeri 1, 2 e 7 sono esauriti, vogliate inviarmi i numeri arretrati .... al prezzo di L. 5.000 cadauno per richieste fino a 4 numeri, o di L. 4.000 cadauno per richieste oltre i 4 numeri arretrati, e perciò per un totale di L............... Sono a conoscenza che i fascicoli suddetti non saranno inviati in contrassegno e, pertanto, ho provveduto oggi stesso a versare il canone di L.................. a mezzo c/c postale n. 37952207 intestato a:
Systems Editoriale - V.le Famagosta, 75 - 20142 Milano

PICCOLI ANNUNCI...

CERCO/OFFRO CONSULENZA...

INVIARE IN BUSTA CHIUSA E AFFRANCANDO SECONDO LE TARIFFE VIGENTI A:

COMMODORE COMPUTER CLUB

V.le Famagosta, 75
20142 Milano

INVIARE TUTTA LA PAGINA ANCHE SE SI UTILIZZA UNA SOLA SCHEDA

Nome ______________ Cognome ______________

Via ______________ N° ______ CAP. ______ Città ______________

Telefono ______________ Orario ______________

S systems